GLOBAL OPTIMISATION OF MULTIPLE GRAVITY ASSIST TRAJECTORIES

MATTEO CERIOTTI

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

Department of Aerospace Engineering
Faculty of Engineering
University of Glasgow

Matteo Ceriotti: *Global optimisation of multiple gravity assist trajectories*
Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy
Department of Aerospace Engineering, Faculty of Engineering
University of Glasgow

ADVISOR:
Dr. Massimiliano Vasile

EXAMINERS:
Dr. Victor M. Becerra
School of Systems Engineering, University of Reading
Reading, United Kingdom

Dr. Euan W. McGookin
Department of Aerospace Engineering, University of Glasgow
Glasgow, United Kingdom

Glasgow, Scotland, United Kingdom
20 May 2010

ABSTRACT

Multiple gravity assist (MGA) trajectories represent a particular class of space trajectories in which a spacecraft exploits the encounter with one or more celestial bodies to change its velocity vector; they have been essential to reach high Δv targets with low propellant consumption. The search for optimal transfer trajectories can be formulated as a mixed combinatorial-continuous global optimisation problem; however, it is known that the problem is difficult to solve, especially if deep space manoeuvres (DSM), are considered.

This thesis addresses the automatic design of MGA trajectories through global search techniques, in answer to the requirements of having a large number of mission options in a short time, during the preliminary design phase. Two different approaches are presented. The first is a two-level approach: a number of feasible planetary sequences are initially generated; then, for each one, families of the MGA trajectories are built incrementally. The whole transfer is decomposed into sub-problems of smaller dimension and complexity, and the trajectory is progressively composed by solving one problem after the other. At each incremental step, a stochastic search identifies sets of feasible solutions: this region is preserved, while the rest of the search space is pruned out. The process iterates by adding one planet-to-planet leg at a time and pruning the unfeasible portion of the solution space. Therefore, when another leg is added to the trajectory, only the feasible set for the previous leg is considered and the search space is reduced. It is shown, through comparative tests, how the proposed incremental search performs an effective pruning of the search space, providing families of optimal solutions with a lower computational cost than a non-incremental approach. Known deterministic and stochastic methods are used for the comparison. The algorithm is applied to real MGA case studies, including the ESA missions BepiColombo and Laplace.

The second approach performs an integrated search for the planetary sequence and the associated trajectories. The complete design of an MGA trajectory is formulated as an autonomous planning and scheduling problem. The resulting scheduled plan provides the planetary sequence for a MGA trajectory and a good estimation of the optimality of the associated trajectories. For each departure date, a full tree of possible transfers from departure to destination is generated. An algorithm inspired by Ant Colony Optimization (ACO) is devised to explore the space of possible plans. The ants explore the tree from departure to destination, adding one node at a time, using a probability function to select one of the feasible directions. Unlike standard ACO, a taboo-based heuristics prevents ants from re-exploring the same solutions. This approach is applied to the design of optimal transfers to Saturn (inspired by Cassini) and to Mercury, and it demonstrated to be very competitive against known traditional stochastic population-based techniques.

CONTENTS

List of Figures

List of Tables

List of Algorithms

ACKNOWLEDGEMENTS

I would like to thank all the members of the SpaceART, Space Advanced Research Team. In particular, Dr. Camilla Colombo, Nicolas Croisard, Dr. Joan-Pau Sanchez, Dr. Christie Maddock, Daniel Novak. With them, I spent most of my time in the last four years, both at work and after, during enjoyable evenings and pleasant weekends. Not only have they been fine colleagues, but also, and most important, good friends.

I am grateful to my advisor, Dr. Massimiliano Vasile, who first of all offered me the possibility, back in 2006, to start a Ph.D. in Glasgow. During my research, his technical advices, inspiration, and trust have been essential for completing this thesis.

I must also not forget other friends in the department: Dr. Gianmarco Radice and Dr. Giangi Gobbi, for all the nights out; David Moore, for teaching me Scots and Scotticisms; Stuart Grey, for adding a bit of Englishness.

Thanks to all the colleagues at the Mission Analysis Office (OPS-GFA) of ESA-ESOC (European Space Agency, European Space Operations Centre) in Darmstadt, Germany. In particular, Johannes Schoenmaekers, who made the internship possible; Dr. Arnaud Boutonnet and Dr. Paolo De Pascale, for their supervision and sharing with me their great expertise in mission analysis; Dr. Elisabet Canalias, for sharing with me her methods and results; Michael Khan, for the enjoyable all-around conversations at coffee breaks; Dr. Rüdiger Jehn, for good suggestions and friendship; Dr. Daniel Garcia, for all the social events. During the three months I spent there, I had great time and an incredibly useful professional experience.

I shall also acknowledge the Advanced Concepts Team (ACT) at ESA, in particular Dr. Claudio Bombardelli, for funding part of this research through the ARIADNA Study 06-4101c “Global Trajectory Optimisation: Can We Prune the Solution Space when Considering Deep Space Manoeuvres?”, in which many ideas were developed. I also thank Dr. Victor Becerra and Dr. Slawomir Nasuto of the University of Reading, for many useful chats and suggestions during the same study.

Finally, I would like to thank my family, who accepted my difficult decision to come to Glasgow, and supported me every time, despite the distance. This thesis is dedicated to them.

Matteo Ceriotti
Glasgow, 20 May 2010

PUBLICATIONS

Some ideas and figures have previously appeared, or will appear, in the following publications:

M. Vasile, M. Ceriotti, P. De Pascale, "An incremental approach to the solution of global trajectory optimization problems", in *Proceedings of Advances in global optimization: methods and applications (AGO 2007)*, Myconos, Greece, 2007.

M. Ceriotti, M. Vasile, C. Bombardelli, "An incremental algorithm for fast optimisation of multiple gravity assist trajectories", in *Proceedings of 58th International Astronautical Congress (IAC 2007)*, Hyderabad, India, 2007.

V. M. Becerra, S. J. Nasuto, J. D. Anderson, M. Ceriotti, C. Bombardelli, "Search space pruning and global optimization of multiple gravity assist trajectories with deep space manoeuvres", in *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, 2007.

M. Vasile, M. Ceriotti, G. Radice, V. M. Becerra, S. J. Nasuto, et al., "Global trajectory optimisation: can we prune the solution space when considering deep space manoeuvres?", European Space Agency, Advanced Concepts Team, http://www.esa.int/gsp/ACT/ariadna/completed.htm#MA, 2008.

M. Ceriotti, M. Vasile, "Automatic MGA trajectory planning with a modified Ant Colony Optimization algorithm", in *Proceedings of 21th International Space Flight Dynamics Symposium (ISSFD 2009)*, Toulouse, France, 2009.

M. Ceriotti, M. Vasile, "MGA trajectory planning with an ACO-inspired algorithm", in *Proceedings of 60th International Astronautical Congress (IAC 2009)*, Daejeon, Republic of Korea, 2009.

M. Ceriotti, M. Vasile, "An ant system algorithm for automated trajectory planning", in Proceedings *of IEEE World Congress on Computational Intelligence (WCCI 2010/CEC 2010)*, Barcelona, Spain, 2010.

M. Vasile, M. Ceriotti, "Incremental techniques for global space trajectory design", in *Spacecraft Trajectory Optimization*, B.A. Conway, Editor. Cambridge University Press, 2010. ISBN 978-0521518505

M. Ceriotti, M. Vasile, "MGA trajectory planning with an ACO-inspired algorithm", accepted for publication in *Acta Astronautica*. DOI: 10.1016/j.actaastro.2010.07.001

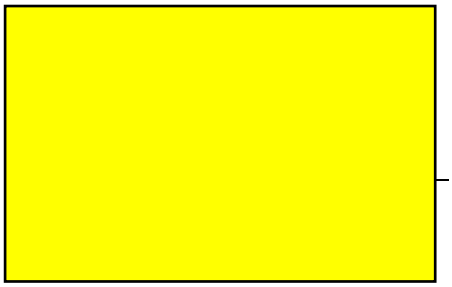
M. Ceriotti, M. Vasile, V. M. Becerra, J. D. Anderson, C. Bombardelli, "An incremental algorithm for the optimization of multiple gravity assist trajectories", submitted to *Journal of Aerospace Computing, Information, and Communication*, AIAA.

M. Ceriotti, M. Vasile, "Automated multigravity assist trajectory planning with a modified ant colony algorithm", accepted for publication in *Journal of Aerospace Computing, Information, and Communication*, AIAA. DOI: 10.2514/1.48448

AUTHOR'S DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Glasgow, Scotland, 20 May 2010



NOMENCLATURE

The author attempted to use standard symbols and acronyms in use in Astrodynamics and Optimisation, giving alternatives wherever appropriate, and tried to stay consistent throughout the document. Some symbols have duplicate meanings, but the appropriate one should be apparent from the context.

Acronyms

AU	Astronomical Unit (149,597,870.7 km)
ACO	Ant Colony Optimization
ACT	Advanced Concepts Team (ESA)
AGA	Aero-gravity assist
AS	Ant System
BS	Beam Search
C	Callisto
Co	Comet
DE	Differential Evolution
DIRECT	Divided Rectangles
DSM	Deep space manoeuvre
E	Earth
EJSM	Europa Jupiter System Mission
EP	Evolution programming
EPIC	Evolutionary Predictive Interval Computation
ESA	European Space Agency
ESOC	European Space Operations Centre (ESA)
GAM	Gravity assist manoeuvre
GA	Genetic Algorithm
GAP	Gravity assist plots
GASP	Gravity Assist Space Pruning
GGA	Ganymede gravity assist (Laplace mission)
GOI	Ganymede orbit insertion (Laplace mission)
GTOC	Global trajectory optimization competition
IMAGO	Interplanetary Mission Analysis Global Optimization
J	Jupiter
JAXA	Japanese Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory (NASA)
LB	Lower bound

LTGA	Low-thrust gravity assist
M	Mars
MA	Memetic algorithm
MATLAB	Matrix Laboratory (® Mathworks)
MESSENGER	Mercury Surface, Space Environment, Geochemistry, and Ranging mission
Mo	Moon
MDTOP	Mission-direct trajectory optimization program
Me	Mercury
MBH	Monotonic Basin Hopping
MGA	Multiple gravity assist
MJD2000	Modified Julian Day since 1st January 2000, 12:00 noon
MS	Multi-Start
MTSP	Motorised travelling salesman problem
N	Neptune
NAIF	Navigation and ancillary information facility
NASA	National Aeronautics and Space Administration
NP	Non-deterministic polynomial-time
OOS	Open shop scheduling
PAMSIT	Preliminary analysis of multiple swing-bys interplanetary trajectories
PSO	Particle Swarm Optimization
RCPSP	Resource-constrained project scheduling problem
S	Saturn
SEPTOP	Solar Electric Propulsion Trajectory Optimization Program
SPICE	Spacecraft planet instrument camera-matrix events (NASA software)
STOUR	Satellite Tour Design Program
TSP	Travelling salesman problem
U	Uranus
UB	Upper bound
V	Venus

Variables

a	Semimajor axis
A, B, C	Coefficients in linear trigonometric equation
b	Identifier of a body or planet
b	Binary variable
$\mathbf{b}$	Box bound
$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$	Unit vectors defining swing-by reference system (Izzo's formulation)
B	Box
C_r	Crossover parameter (DE)

$\mathbf{d}$	Discrete probability distribution vector (ACO-MGA)
D	Domain or search space
$\bar{D}$	Residual domain after pruning
e	Eccentricity
E	Orbital energy
E	Eccentric anomaly
f, g	Generic functions
$\bar{f}$	Pruning threshold
$f_{1/2}$	Binary variable for intersection point (ACO-MGA model)
$f_{p/a}$	Binary variable for pericentre or apocentre (ACO-MGA model)
F	Perturbation parameter (DE)
g	Generation (modified MACS)
G	Universal constant of gravitation ($6.6695 \cdot 10^{-11}$ $\mathrm{m}^3/\mathrm{kg} \cdot \mathrm{s}^2$)
$\mathbf{G}$	Cartesian product matrix (ACO-MGA)
$\hat{\mathbf{h}}$	Out-of-plane unit vector
i	Inclination
i, j, k	Generic indexes
I	Interface (block model)
h, k	Discrete values
$\mathbf{K}$	Vector of 6 Keplerian parameters
l	Generic index for level or leg
L	List
m	Mass
m_{DSM}	Magnitude of deep space manoeuvre (ACO-MGA model)
M	Deep space manoeuvre point
n	Generic number
$\hat{\mathbf{n}}$	Normal unit vector
$\mathbf{n}_r$	Reference unit vector (swing-by)
$\mathbf{n}_\Pi$	Unit vector normal to swing-by plane
n_{arcs}	Number of arcs
$n_{back,max}$	Maximum number of backward legs
$n_{backspacing,max}$	Maximum body spacing in list for two consecutive swing-bys
n_{DSM}	Number of deep space manoeuvres
n_{eval}	Number of evaluations
n_{iter}	Number of iterations
n_{legs}	Number of legs
n_{nodes}	Number of nodes
n_{pop}	Size of population (modified MACS)
$n_{popratio}$	Number of selected agents (modified MACS)
n_{rev}	Number of revolutions

n_{runs}	Number of runs
n_P	Number of planets
$n_{rsb,max}$	Maximum number of resonant swing-bys
n_{sb}	Number of swing-bys
n_{trials}	Number of trials
$\mathbf{n}_{par}$	Vector with numbers of each parameter
N_L	Number of levels
p	Number of sub-intervals or subdivisions
p	Index of run
p	Orbit parameter
P	Generic planet or body
P	Orbital period
P	Population (modified MACS)
Pr	Probability (ACO-MGA)
q	Number of boxes
q_{left}	Number of remaining boxes
$\mathbf{q}$	Ordered sets with possible parameters (ACO-MGA)
r, s	Shaping parameters
r_p	Radius of pericentre (hyperbola of swing-by phase)
r_π	Radius of pericentre (Tisserand plane)
r_{ps}	Signed radius of pericentre (ACO-MGA model)
R_P	Mean radius of planet
$\mathbf{r}$	Position vector
$\hat{\mathbf{r}}$	Radial unit vector
$\mathbf{s}$	Planetary sequence
t	Time
$\hat{\mathbf{t}}$	Tangential unit vector
t_0	Initial (or launch) time
T	Time of flight
tol_{conv}	Tolerance on convergence (modified MACS)
tol_f	Tolerance on global optimality
u	Bound of basin
$\mathbf{v}$	Velocity vector
$\mathbf{v}_0$	Launch excess velocity (relative to planet)
v_e	Exhaust velocity of propellant mass
v_∞	Hyperbolic excess velocity
V	Volume
w_{planet}	Weight on planet selection (ACO-MGA)
w_{type}	Weight on type of transfer selection (ACO-MGA)

$\mathbf{x}$	Solution vector
X	ε -set
y	Objective value
α	Fraction of time of flight at which the DSM occurs
β	Weight
γ	Attitude of the swing-by hyperbola plane
δ	Deflection angle (swing-by)
δ	Launch declination
$\overline{\delta}$	Non-dimensional, uniform launch declination
δ_f	Distance from global optimum
Δ	Discriminant
Δv	Change in velocity
Δ_θ	Small angular displacement
ε	Threshold of the ε -set
ζ	Attitude of swing-by plane (Izzo's model)
θ	In-plane angle (right ascension)
θ	True anomaly
$\overline{\theta}$	Non-dimensional, uniform in-plane angle (right ascension)
$\hat{\theta}$	True anomaly of spacecraft on planet's orbit at intersection time
$\hat{\boldsymbol{\theta}}$	Transversal unit vector
λ	r_{ps} or v_0 (ACO-MGA model)
Λ	Distance between planet and hyperbola asymptote (swing-by)
μ	Planetary constant (mG)
ν	$\tan(\theta/2)$
π	Greek Pi (3.1415…)
Π	Swing-by plane
ρ_l	Radius of neighbourhood (modified MBH and modified MACS)
$\boldsymbol{\tau}$	Pheromone distribution vector
υ	Number of intervals on each dimension
φ	Out-of-plane angle (declination)
φ_0	In-plane launch direction (ACO-MGA model)
ϕ	Difference in orbit orientation (ACO-MGA model)
Φ	Pruning criterion
$\boldsymbol{\psi}$	Two criterion vector (modified MACS)
ω	Rotation of line of apsides (in powered swing-by)
Ω	Right ascension of ascending node

Subscripts and Superscripts

$\square_0$	Referred to departure or launch
$\square_1$	Generic initial state
$\square_2$	Generic final state
$\square^{(1)}, \square^{(2)}$	Referred to intersection point 1 or 2 (ACO-MGA model)
$\square_{abs}$	Absolute
$\square_c$	Candidate
$\square_d$	Discontinuity
$\square_{DSM}$	Referred to deep space manoeuvre
$\square_{est}$	Estimated
$\square_f$	Final
$\square_{feas}$	Feasible
$\square_h$	Out-of-plane component
$\square_{inc}$	Incomplete
$\square_{int}$	Intersection
$\square^{(int)}$	At selected intersection
$\square_l$	Lower bound
$\square_{L,i}$	Related to level i only
$\square^{(i)}$	At i^{th} step (in algorithms)
$\square_{min}$	Minimum
$\square_{max}$	Maximum
$\square_M$	Referred to a point in the deep space or the manoeuvre in it
$\square_P$	Referred to planet
$\square_T$	Referred to type of transfer (ACO-MGA)
$\square_\theta$	Transversal component
$\square^-$	Incoming
$\square^+$	Outgoing
$\square^*$	Optimal
$\tilde{\square}$	In the affine space, non-dimensional
$\hat{\square}$	Unit vector
$\overline{\square}$	Residual after pruning, or pruning threshold
$\square_\infty$	At infinity
$\square'$	Referred to the case with velocity overturning (swing-by)

Operators

$\lvert\square\rvert$	Norm of a vector, cardinality of a discrete set, dimensionality of a real-valued set
$\dot{\square}$	Differentiation with respect to time
$B(\square)$	Behaviour function (modified MACS)
$d\square$	Differential
$I_d(\square)$	Dominance index (modified MACS)
$N(\square)$	Neighbourhood of a given point (modified MBH)
$\mathrm{Vol}(\square)$	Volume of a set
$\Delta\square$	Generic variation of a quantity
$\times$	Cartesian product
$\succ$	Dominance
$\wedge$	And

1

INTRODUCTION

This chapter introduces multiple gravity assist (MGA) trajectories and their optimal design. First, a brief overview of past, present and future MGA missions is given. This is followed by an overview of classic and modern techniques for mission design, including some based on global optimisation techniques. Then, the motivations and objectives of this work will be presented.

1.1 Space Exploration

Space exploration has always fascinated the human mind: the roots of men's desire to lift from Earth are to be found in ancient times, when men were impressed by the presence of other bodies floating in the sky. The sky, stars and planets represented the object of their observation, and methods and instruments kept evolving and improving from then on. The first great step, which provided a connection between the Earth and the universe, was the invention of the refracting telescope: at the beginning of the 17th century, Galileo was the man who improved it at a point that the rings of Saturn and the moons of Jupiter could be clearly seen [1]. Since then, the images of the universe got more and more focused and accurate. The initial fascination turned into studying the possibility for human beings to explore celestial bodies more closely. The first important work that theorizes this possibility was written by the Russian scientist Konstantin Tsiolkovsky in 1903 [2]. Consequently, during the first half of the 20th century many studies were dedicated to the building of solid and liquid-propellant rockets; in particular, the German V-2 rocket [3] is the first human artefact to achieve sub-orbital spaceflight: the year was 1942.

Floating in outer space remained a dream for men at that time, but became a reality for animals; that was due to the necessity of testing the survivability outside the Earth's atmosphere and with no gravity. Between 1940 and 1960, fruit flew at first, then monkeys, mice and dogs were launched into space [4], particularly by the United States and USSR. The dog Laika acquired a peculiar mention for being the first animal in orbit. Yet Laika died during the flight and many other similar tests

were necessary to grant the success of Yuri Gagarin's mission: on 12 April 1961 he reached the outer space and orbited the Earth [5].

That decade saw the greatest step in the history of space exploration, mainly pushed by the space race between USA and USSR; the eight years after the first human being in space prepared the approach to another celestial body. The Moon became the first destination to be explored beyond the Earth's atmosphere: on 21 July 1969 American Neil Armstrong and Edwin E. Aldrin set foot on the Earth's satellite on a mission aboard Apollo 11 [6].

The exploration of the universe around our world had just begun; from 1970s flybys and orbital flights were conducted around planets such as Venus [7], Jupiter [8], Mercury [9], Saturn and, above all, Mars [10-14]. This last one seems to be the next space destination where to put the terrestrial flag [15, 16].

Although there is no plan to send human beings to farther planets, still their robotic exploration is of great interest, for scientific reasons. Unfortunately, reaching Mercury, Jupiter or the outer part of the solar system, is much more difficult than reaching Mars. Current launch and propulsion capabilities are not sufficient to deliver the required payload to those destinations: for example, the mass of propellant that yields the required change in velocity for an Earth-Saturn Hohmann[1] transfer is 99% of the mass of the spacecraft, considering current chemical engine technology. It is evident that the amount of propellant to deliver a payload and the subsystems to make it functional could be enormous. For this reason, gravity assist manoeuvres have been devised to gain the required change in velocity with no propellant consumption.

1.2 Multiple Gravity Assist Missions

A *gravity assist manoeuvre* (GAM, also called *swing-by* or *gravitational slingshot*), is the use of the relative movement and gravity field of a planet or other massive celestial body to change the velocity of a spacecraft [17]. This is achieved with a close proximity swing-by of the celestial body so that its gravity produces a change in the velocity vector of the spacecraft. The complete mathematical formulation of a swing-by will be presented in detail in Chapter 2. For the moment, we just need to imagine that the spacecraft during its interplanetary journey encounters a massive body with some relative speed. Due to this "close passage", there is a momentum exchange between the spacecraft and the body, so that the spacecraft increases, decreases or rotates its inertial velocity. The body loses a very small proportion of its orbital momentum due to the significant mass compared to that of the probe. However, the change in velocity of the spacecraft can be significant.

If no gravity assists are used throughout an interplanetary journey, the whole velocity change (Δv) needed to go from the Earth's orbit to the target orbit shall be provided by a propulsion system. Usually the launcher gives the first part of the velocity change, then an upper stage provides the velocity necessary to leave the

[1] The Hohmann transfer is the bi-impulse direct transfer between circular, coplanar orbits with minimum change in velocity.

Earth's sphere of influence, and then the spacecraft's thruster completes the transfer by providing the Δv to complete the journey and possibly to insert into the target orbit, if any.

In traditional propulsion systems, the change in velocity (e.g. acceleration) is caused by the ejection of an accelerated propellant mass. Therefore, the available impulse is limited by the propellant mass on-board. Thus saving on propellant mass means either a lighter spacecraft, or more mass available for the payload. In turn, a lighter spacecraft implies a cheaper launch (the launch has a cost per kilogram of mass), or a launch into a higher energy orbit. Other innovative propulsion systems exist, like solar sails [18], that do not consume any mass. The current technology, though, poses some limitations in constructing large, deployable sails, and therefore they have not been used yet for interplanetary travel.

Gravity assists have been exploited in the past 40 years to reach targets with very high or very low energy orbits with respect to the Earth, or even to considerably change the heliocentric orbit inclination. All these types of orbits are known under the category of high-Δv targets, due to the high Δv budget necessary to reach them from Earth.

1.2.1 Past, Present and Future Missions

The importance of gravity assist manoeuvres is testified by the high number of missions which have made use of this enabling technology in the history of interplanetary space exploration. This section contains an overview of the past and current interplanetary missions that exploited one or more gravity assist. In addition, two very ambitious MGA missions, currently under study, will be presented.

The Mariner 10 probe [9] was the first spacecraft to use the gravitational slingshot effect to reach another planet. In fact, one of the objectives of this NASA mission was to prove the possibility of using gravity assist manoeuvres. The probe swung by Venus on February 5, 1974, on its way to becoming the first spacecraft to explore Mercury.

Pioneer 11 [8] was the second mission of the Pioneer program to investigate Jupiter and the outer solar system (after its sister probe Pioneer 10) and the first to explore Saturn and its main rings. Launched in 1973, Pioneer 11 used Jupiter's mass in a gravity assist to get the energy to reach Saturn. The spacecraft made a successful swing-by of Saturn and then followed an escape trajectory from the solar system. The interplanetary trajectories of Pioneer 10 and 11 are shown in Fig. 1.1.

After the success of Mariner and Pioneer, NASA decided to continue the interplanetary exploration programme with the Voyager 1 and Voyager 2 probes [19], with the ambitious aim of including MGAs in the same mission. Although they were originally designated to study just Jupiter and Saturn, the two probes were able to continue their mission into the outer solar system. Both probes were injected into a direct transfer to Jupiter in 1977. Voyager 2 was launched first, and encountered Jupiter, Saturn, Uranus and Neptune (sequence EJSUN). Voyager 1 was instead launched on a faster trajectory, which enabled it to reach Jupiter and Saturn sooner at the consequence of not visiting the outer planets. Both spacecraft

are currently on course to eventually exit the solar system, and Voyager 1 is currently the human made object farthest from Earth (see Fig. 1.2).

The two Voyager missions were made possible by a very favourable alignment of the outer planets (Jupiter, Saturn, Uranus and Neptune), which happened from 1970 to 1990: a similar alignment will not occur again until the middle of the 22nd century. This is the reason why the trajectories of the two probes were named "Grand Tour" [20]. In fact, the main design constraint of a MGA trajectory is to have all the celestial bodies in the right place at the right time. This is known as the "phasing problem", and, throughout this work, we will see how to address it and when to neglect it.

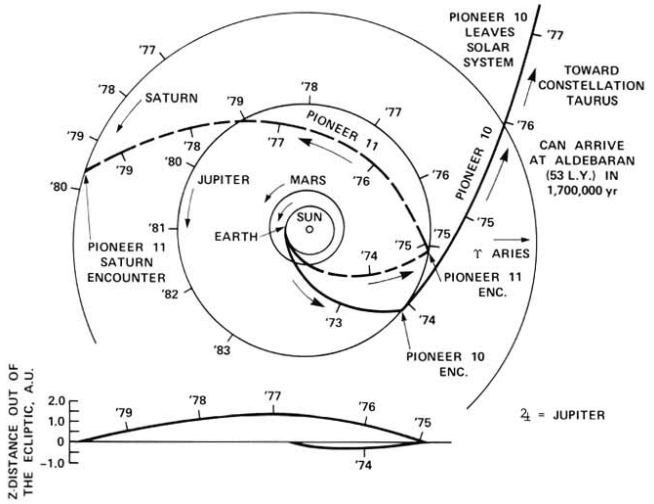


Fig. 1.1. Pioneer 10 and 11 trajectories (credit: NASA).

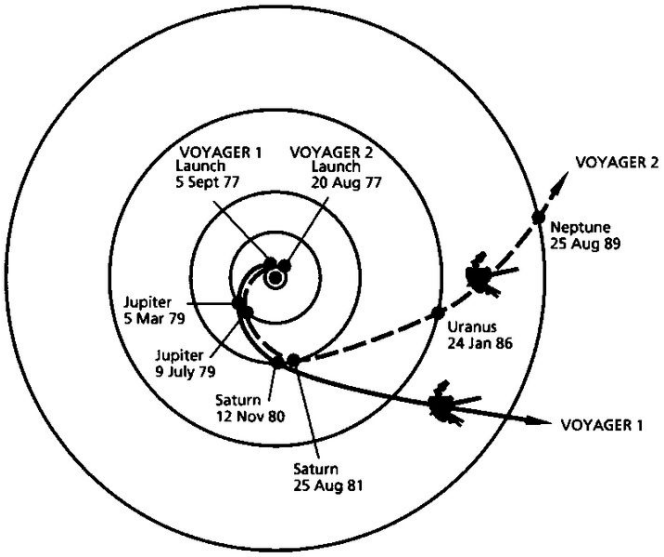


Fig. 1.2. Voyager 1 and Voyager 2 trajectories (credit: NASA).

The Galileo [21] spacecraft was originally designed to be injected into a direct transfer to Jupiter from the space shuttle. New safety protocols introduced as a result of the Challenger accident, in 1986, forced Galileo to use a lower-powered upper stage booster rocket, which could not provide the necessary energy. The trajectory of Galileo was re-designed, including a swing-by of Venus and two swing-bys of the Earth, allowing the probe to reach Jupiter in 1995. After injection, the spacecraft travelled around Jupiter in elongated elliptical orbits, designed for close up fly-bys of Jupiter's largest moons. The successful mission was even extended to perform a number of fly-bys of Europa and Io.

The trajectory of Cassini-Huygens [22-24] is the most complex MGA trajectory designed for a mission to an outer planet. The aim of the mission was to study Saturn and its complicated planetary system of satellites. The mass of the spacecraft at launch was an impressive 5600 kg, of which more than 3000 kg of propellant, needed for deep space manoeuvring and for the final injection around Saturn. Due to the high mass budget of the spacecraft, the launch hyperbolic escape velocity was very low (below 4 km/s). Therefore a complex MGA trajectory was designed: the spacecraft was launched in 1997 and entered into orbit around Saturn in 2004, exploiting two gravity assists of Venus, one of the Earth, and one of Jupiter (sequence EVVEJS, Fig. 1.3). The Cassini probe, once orbiting around Saturn, made also use of a huge number of swing-bys of several moons to modify its orbit in several ways, such to study the Saturn system in the most complete way possible.

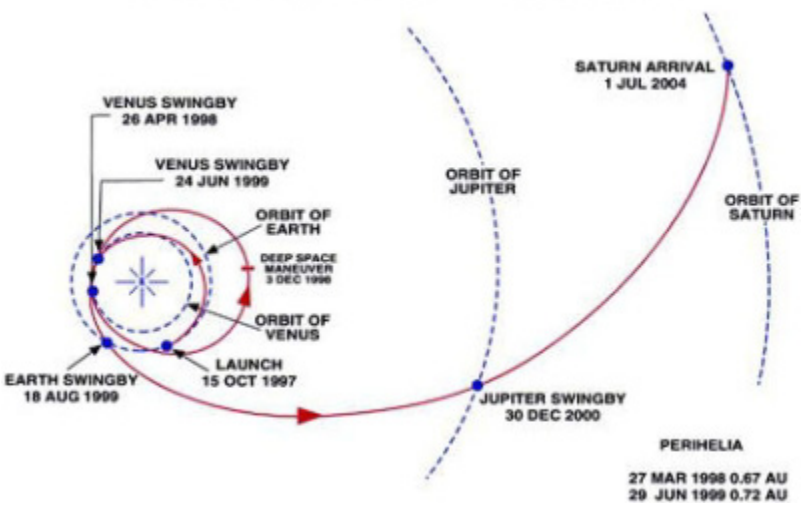


Fig. 1.3. Cassini trajectory (credit: ESA/NASA).

The second mission to Mercury, NASA's MESSENGER [25, 26], launched in 2004, also made use of a swing-by of the Earth and two additional swing-bys of Venus to reach its target. It also performed three swing-bys of Mercury to lower its relative velocity and ease the orbital insertion manoeuvre, which is scheduled for 2011. The complete planetary sequence is then EEVVMeMeMeMe (see Fig. 1.4).

For this mission, the gravity assist manoeuvres were used to both change the in-plane velocity components of the spacecraft (energy change), and to change the

orbital inclination, as the orbit of Mercury has an inclination of about 7 degrees over the ecliptic.

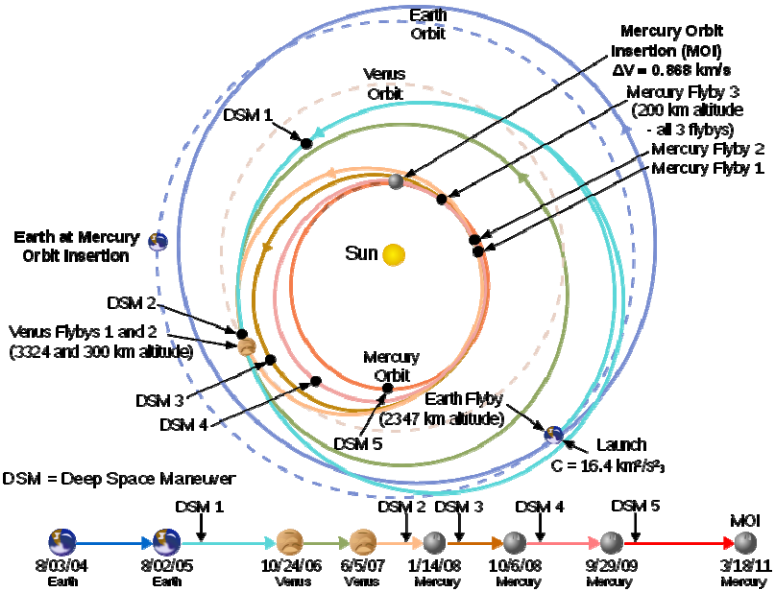


Fig. 1.4. MESSENGER trajectory (from [25]).

The European Space Agency (ESA) Rosetta [27] probe was launched in 2004, with the scope of rendezvousing the comet 67P/Churyumov-Gerasimenko, and releasing a lander, Philae, on it. Similar to Cassini, this mission exploited a rather complicated sequence of gravity assists perform the rendezvous with minimum propellant mass. Rosetta used an Earth swing-by, a Mars swing-by, followed by two more Earth swing-bys. The rendezvous with the comet is scheduled for 2014 (sequence EEMEECo, Fig. 1.5).

A fast trajectory (Fig. 1.6), exploiting only one gravity assist (of Jupiter), was designed, instead, for the NASA New Horizons [28] mission to Pluto and the Kuiper belt. The spacecraft was launched in 2006, and the Jupiter swing-by happened in 2007. Despite the fast hyperbolic heliocentric trajectory, the arrival at Pluto is expected to occur in 2015, and in the following years the probe will cross the Kuiper belt.

The missions described so far performed gravity assist manoeuvres to reach destinations with a small orbital inclination difference with respect to the ecliptic. The gravity manoeuvres were used to change mainly the energy of the orbit. Other missions, instead, used the gravitational slingshots to change the orbital plane. The space probe Ulysses [29], designed by NASA and ESA to study the solar poles and launched in 1990, exploited a single swing-by of Jupiter, in 1992, to increase the inclination to the ecliptic by 80.2 deg.

Other MGA missions are currently under investigation by the major space agencies.

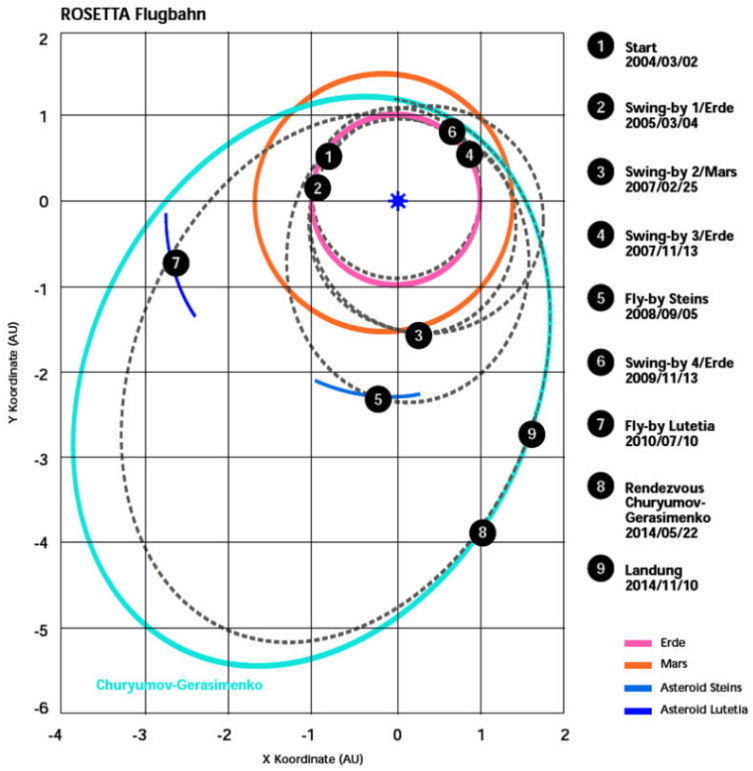


Fig. 1.5. Rosetta trajectory (credit: http://www.enterprisemission.com).

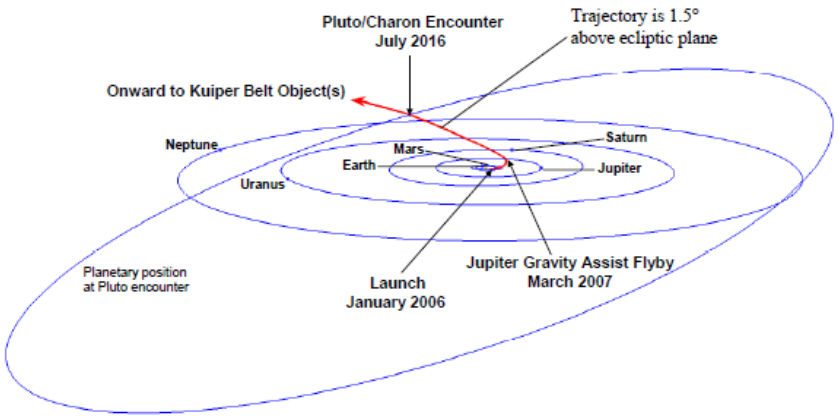


Fig. 1.6. New Horizons interplanetary trajectory (from [28]).

BepiColombo [30-34] is the ESA cornerstone mission to Mercury, in partnership with JAXA (Japan Aerospace Exploration Agency). Several options have been studied in the last 10 years. All of them include one or more swing-bys of the Earth, Venus and Mercury. Some include also one or more swing-bys of the Moon. An example showing a MoEVVMeMeMeMe sequence can be seen in Fig. 1.7. Two orbiters and a transfer module, consisting of electric propulsion and

chemical propulsion units, will be launched as a single composite spacecraft. The expected launch date is 2013, and arrival at Mercury is in 2019.

The total transfer time is about six years, two of which are spent before the first encounter of Mercury. As a comparison, a direct Hohmann transfer to Mercury takes only 105 days. A long mission time, and the need for an accurate targeting of the celestial bodies during swing-by manoeuvres demand for long and precise operations. Therefore, the gain in propellant offered by gravity assist manoeuvres comes at the cost of a higher operation cost. A trade-off between mission time and propellant mass is always required. In this sense, designing an MGA trajectory is analogous to solving a multiple objective optimisation problem, in which the two objectives are partially conflicting. In this work, it will be shown how to take into account the two merit functions.

ESA, NASA and JAXA are also studying a mission to Jupiter and the Jovian moon system. The mission is currently known as Europa Jupiter System Mission (EJSM) or Laplace [35]. With the ambitious goal of determining whether the Jupiter system harbours habitable worlds, the mission consists of two separate spacecraft, possibly to be launched in 2020. Both spacecraft will use Venus, Earth, Earth gravity assists to reach Jupiter (sequence EVEEJ). The mission will then continue around Jupiter with multi-year tours of the Jovian system, including many flybys of Io, Europa, Ganymede and Callisto. Finally, the two probes will inject around Europa and Ganymede respectively. For a more detailed overview of the phases of this mission, see Section 4.2.

These last two missions will be used as case studies in this work.

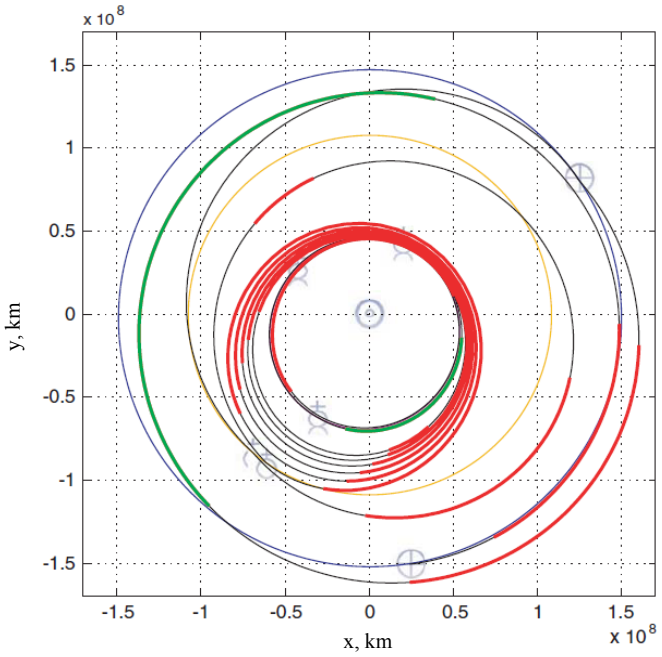


Fig. 1.7. One of the trajectories investigated for BepiColombo. Red and green arcs are thrusting arcs. From [32].

1.2.2 Classic Design Techniques

As the overview of the missions demonstrates, MGA trajectories have been extensively investigated over the last forty years. Especially in the early times, their preliminary design was approached mainly relying on the intuition of mission designers in unison with analysis tools, based on a number of simplifying assumptions, such as the Tisserand's graph [36, 37]. This tool will be presented in detail later in this work (see Appendix E for an overview and Chapter 4 for an application); for now it will suffice to say that it allows the definition of possible gravity assist sequences in a given planetary system.

However, graphical tools, like the Tisserand's graph, are unsuitable to design transfers to highly eccentric or inclined orbits. Moreover, they neglect the phasing problem, and thus do not give any information on the departure time. Also, they cannot be used to plan intermediate propelled manoeuvres or Δv-GAM transfers.

All these techniques were mainly used to scan a large range of alternative options in order to identify a restricted number of potentially good solutions (or first guesses).

The accurate design of these solutions was performed as a second step, by using higher fidelity models, local optimisation techniques, and optimal control theory [38], but the use of these techniques was constrained by the limited computing power available at that time. However, any iterative local solution method based on recursive formulas (e.g. Newton or quasi-Newton methods) requires a first guess solution. The choice of the first guess affects the convergence of the method and the quality of the optimised solution. It follows that the first step in designing a trajectory, i.e. the identification of a first guess, has a great impact on the final solution.

In addition, even for simple planet-to-planet transfers, there could be points of discontinuity, non-differentiability and multiple local minima [39-41]. All these features of the search space can make the identification of a globally optimal solution problematic. Note that, although the global optimum is not generally needed to satisfy mission requirements, its identification would be useful to set a threshold on the best achievable result.

1.2.3 Multi-Impulse Trajectories

A common approximation that was (and still is) used to design interplanetary trajectories is that of considering high-thrust arcs, as instantaneous velocity changes, or impulses: i.e., to assume that the engine can provide an amount of thrust such that the velocity vector changes in a negligible time. When these manoeuvres happen within a planet-to-planet leg, they are named deep space manoeuvres (DSM).

Although this approximation simplifies considerably the problem, it was shown that the complexity of the search space increases consistently when DSMs are considered along a MGA trajectory: the objective function presents a higher number of local minima. This phenomenon is deeply studied and clearly shown in [41]. The authors consider two test cases: a direct planet to planet transfer, and a transfer

through a gravity assist. In both cases, first the ballistic case is considered, then a DSM is inserted either between the planets or after the swing-by. The result is that the DSM increases the number of optimal feasible paths. Although the globally optimal launch window is still the same, the number of opportunities for that window has increased and the distribution of local minima becomes almost continuous over a wide set of launch dates. The authors conclude that, whereas an increase in the degrees of freedom due to the DSM could be somehow expected, less obvious is the benefit in terms number of optimal solutions resulting from the increased complexity and multimodality of the problem.

This once again highlights the importance of the selection of one or more first guesses, before the local optimisation step.

1.2.4 New Trends

The modern approach to space mission design steps through phases of increasing complexity, the first of which is always a mission feasibility study. In order to be successful, the feasibility study phase has to analyse, in a reasonably short time, a large number of different mission options. Each mission alternative can serve as a first guess for more detailed and sophisticated analyses, at a later design phase, and each mission option requires the design of one or more optimal trajectories.

Therefore, it would be desirable to automatically generate many optimal or nearly optimal solutions, over the range of the design parameters (escape velocity, launch date, time of flight, etc…), that are accurate enough to allow a correct trade-off analysis. In other words, the decision maker should be presented with as many options as possible, with each option accurate enough to make the correct decision.

In the last two decades, different methods and tools have been proposed for the automatic design of MGA transfers. Most of these tools are based on systematic search procedures (e.g. STOUR and GASP, which will be covered in the following sections). Others, instead, formulate the problem as a global optimisation or as a global search for multiple local minima.

1.3 Global Optimisation for Trajectory Design

In recent times there has been a flourishing interest in methods and tools for preliminary mission analysis and design, ranging from low-thrust trajectory design [42, 43], to perturbed geocentric orbits [44], to MGA trajectories [45-47]. Also due to the increase of computational power available in personal computers, about twenty years ago, global optimisation techniques started to be extensively used towards the solution of complex interplanetary trajectory transfers. Methods including genetic algorithms [48], neurocontrollers [49], shooting [50], and collocation [51] have been used with varying effectiveness. As shown in [39], the efficiency, both computational and performance-wise, of these approaches are strongly linked to the type of problem that has to be solved.

In the last decade, different forms of stochastic search methods have been applied to orbit design, starting from the work of Hartmann et al. [43] on the use of

global optimisation methods for trajectory design. They proposed the use of a multi-objective, non-dominated sorting genetic algorithm (NSGA) to find planet-to-planet transfers using low thrust propulsion. Classical calculus of variations (implemented in the software SEPTOP) was used to compute the final mass, given an initial estimate of the costates (Lagrange multipliers). NSGA evolved a population of individuals, representing possible trajectories, whose chromosomes were encoding the initial values of the costates and the total time of flight. The paper presented some examples of Pareto-optimal trajectories from the Earth to Mars transfer, in the cases of close approach or rendezvous.

In 2003, Vasile [52] proposed a stochastic global optimiser (EPIC) that was tested on interplanetary transfer, and later used by Vasile and De Pascale as a component of the interplanetary design tool IMAGO [41]. EPIC and IMAGO will be covered in more detail in the next Section 1.4, dedicated to the automatic design of MGA trajectories.

More recently, the scientific community focused on hybridising different global optimisation techniques. Rosa Sentinella [53] showed that the integrated use of Differential Evolution (DE) [54, 55] and genetic algorithms (GA) in a multi-population optimisation procedure can be effective.

In 2008, Vasile et al. [56] proposed a generalisation of Differential Evolution and Particle Swarm Optimization (PSO) [57] in the form of discrete-time dynamical system. The analysis of the local convergence of the dynamical system led to the development of a restart procedure analogous to the one implemented in Monotonic Basin Hopping (MBH) [58]. The new algorithm performed better than either standard DE or MBH on some space trajectory optimisation problems.

The interest towards the global optimisation of interplanetary trajectories is also confirmed by the initiatives of the European Space Agency on this subject.

In 2004, a study was proposed by the ESA Advanced Concepts Team (ACT), to create a taxonomy on global optimisation methods for trajectory design. The study was performed independently by Di Lizia and Radice [39, 40], and Myatt, Becerra et al. [59]. Different types of interplanetary transfer were defined and formalised as box-constrained global optimisation problems, and classified according to the propulsion system (impulsive high-thrust and low-thrust), the dynamic model (two-body dynamics, *n*-body dynamics) and the number of planetary encounters (planet-to-planet, MGA). In particular, for the MGA case, the authors focused on some variants of the Cassini mission. Planetary sequences EMJS, EVEJS and EVVEJS were considered for transfers with powered swing-bys (i.e. a propelled manoeuvre is performed at the pericentre of the swing-by hyperbola). The morphology of the search space of each problem was studied in order to identify typical features which could mostly affect the global search (presence of multiple local minima, size of the basins of attractions, discontinuities, etc.), and complexity analysis of each problem was performed. Finally, the study put to the test eleven global optimisation algorithms, taken from three main classes: stochastic, deterministic and meta-model based. The optimisation problems were submitted to the whole set of optimisation algorithms in the attempt to match the heuristics implemented in each algorithm with the structural properties of each problem.

In 2005 the ACT opened a call for an international Global Trajectory Optimization Competition (GTOC) [60]. The competition aimed at finding, in one month time, the best solution to a given interplanetary optimisation problem. The scientific community responded with enthusiasm and participation and since then, four GTOCs have been organised by the winners of each edition.

The challenge posed to the international scientific community through the 1st ACT Global Trajectory Optimization Competition (GTOC1) problem [60] was to find the best trajectory to deflect the asteroid 2001 TW229 by a direct impact. A proportioned use of both planetary swing-bys and low-thrust could be used to achieve the best impact conditions at the asteroid. Therefore, the search for the sequence of planetary swing-bys, together with finding a first guess of the trajectory, played the most important role on finding a good solution to this problem.

For the solution of each of the four GTOC problems, a multitude of different approaches and techniques were proposed by the participants. However, all the methods were based on intuition and *ad hoc* considerations, together with the utilisation of a large amount of computing power.

1.4 Automatic Design of MGA Trajectories

The automatic design of MGA trajectories is an interesting problem from an operational research point of view. In fact, the number of alternative paths grows exponentially with the number of planetary encounters, and the number of local optima multiplies with the number of gravity assist manoeuvres [39, 41, 61].

Moreover, if DSMs are inserted between two planetary encounters, the number of alternative paths further multiplies times the exponential of the number of DSMs. Thus, the systematic scan of all possible trajectories in a given range of launch dates becomes quickly computationally intensive even for moderately short sequences of gravity assist manoeuvres and short launch windows.

The search for the best transfer trajectory can be formulated as a global optimisation problem, an instance of which is identified by a particular combination of trajectory model, sequence of planetary encounters, boundaries of the search space and optimality criterion. Thus, a different trajectory model would correspond to a different instance of the problem even for the same destination planet and sequence of planetary encounters. In this work it will be shown that some models can make the problem solvable in polynomial time while others make it NP-hard.

In the literature on MGA trajectories, their automatic complete design (i.e. the definition of an optimal sequence of planetary encounters and the definition of one or more locally optimal trajectories for each sequence) has been approached with several different techniques: from deterministic approaches [20, 45], to graphical tools [36, 37], from stochastic approaches [62-65] to hybrid methods [41, 61, 63, 66]. For each approach different modelling of the trajectory has been considered, from reduced two dimensional models [45] to three dimensional models with no DSMs and powered swing-bys [59] to complete three dimensional models with

DSMs [20, 41]. Some examples also exist of the automatic selection of the optimal sequence of planetary encounters [41, 67-69].

All of them can be classified in two main categories: two level approaches, integrated approaches.

1.4.1 Two-Level Approaches

Two-level approaches split the problem into two sub-problems which lay at two different levels: one sub-problem is to find a suitable set of sequences of planetary encounters; the other is to find at least one optimal trajectory for each sequence. Two-level approaches define the planetary sequence independently of the trajectory itself [1]. They use a simplified, low fidelity, model at the first level to quickly assess many, if not all, sequences and a more accurate model, at lower level, to optimise the trajectory. Each sequence is represented by a string of integer numbers, while the associated trajectory is represented with a string of mixed real and integer numbers defining all the characteristics of the events occurring along the trajectory (e.g. launch, DSM, arrival at a celestial body, number of revolutions around the Sun, etc.).

The following sub-sections will focus on tools, based on the two-level approach, that have been developed in recent times.

STOUR

The software tool STOUR (Satellite Tour Design Program), example of two-level approach, was initially implemented at JPL for the design of the Galileo mission. Starting from 1990, the tool was enhanced and extended by Longuski et al. [20] at Purdue University, and further developed at JPL. This tool has been extensively used for the preliminary investigation of interplanetary trajectories to Jupiter and Pluto [70], for the design of the tour of Jovian moons and for Earth-Mars cycling trajectories.

In its first implementation, STOUR was essentially a systematic search of patched conics gravity assist trajectories. The user was first selecting a launch date and a launch excess velocity v_0. The interplanetary leg was a ballistic transfer computed with a Lambert's problem. The user provided the number of full revolutions, and a range of discrete the flight times was considered. The upper and lower bounds of the time of flight were computed considering a multiple of the number of revolutions required, and of the period of the target planet. For each discrete value of the time of flight, and through the ephemerides, all the possible transfers to the next body were computed. For the following legs, which were departing with a swing-by, the time of encounter and the asymptotic velocity v_∞ were given by the arrival conditions of the trajectory at the previous leg. Two legs were then matched at a planet with an unpowered swing-by, with what was called by the authors the "Vis Viva matching problem". Basically, since the swing-by did not change the modulus of the v_∞, two interplanetary legs could be matched at the planetary encounter only if they had the same value of v_∞ at the same time. In the case of departure, the v_∞ was set by the launch. Therefore, by varying the time of

flight of the legs, it was possible to find the values for which the "Vis Viva" was matching at a given planet. Once a match was found, the existence of a trajectory was determined by comparing the incoming and outgoing $\mathbf{v}_\infty$ vectors. In fact, the angle δ between the two was limited by the minimum swing-by altitude through an analytic relationship: if this constraint was satisfied, then a solution existed, and the search could continue by adding further legs to the trajectory. The systematic scan generated the solutions by scanning all the possible planetary sequences and time of flights for a given set of departure conditions. The algorithm could then be run for several launch dates within a launch window.

Two main limitations of this type of approach can be identified: firstly, the systematic search can generate a very high number of possible trajectories to be explored, despite the matching constraints. Secondly, the analysis is limited to purely ballistic trajectories. This is not a big issue for interplanetary tours in the outer solar system, since the great part of the Δv is provided by the launch and the gravity assist manoeuvres, and the DSMs have only the role of providing minor corrections to the trajectory. On the other hand, DSMs become quite important for tours of the inner solar system (towards Mercury), in which in most of the cases, multiple revolutions and resonant[2] swing-bys are used. The same applies for a tour of the Jovian moons, for example. In fact, the DSM in a resonant transfer has the very important role of changing slightly the point in which the spacecraft re-encounters the planet: this yields a considerable change in the relative velocity, thus enhancing the effect of the swing-by.

An additional issue which affects all the approaches based on a systematic search based is the granularity of the search grid: basically, the grid should be thick enough to identify all the promising solutions (fine tuning could be achieved in a second step through classic optimisation), but at the same time it cannot be too thick, as it heavily affects the computational time. Often, deciding the granularity of this grid is not an easy task.

The two limitations of STOUR were overcome by following studies and developments. Starting from 1993, studies by Patel and Longuski [71] and Sims et al. [72], lead to a new version of STOUR, that introduced the possibility of having three different types of Δv manoeuvres: powered swing-by, broken-plane and v_∞-leveraging. In particular, the v_∞-leveraging is a manoeuvre in which a DSM is performed near the aphelion or perihelion during the resonant orbit, such that the spacecraft reencounters the planet with a higher v_∞ for the following gravity assist. This kind of trajectory can be used for example when launching from the Earth towards Venus. It is well known that Venus can be used for one or multiple swing-bys for reaching Mercury or outer planets. Some propellant can be saved if, instead of launching from Earth to Venus, a resonant swing-by of the Earth is performed before reaching Venus, together with the appropriate DSM. The v_∞-leveraging manoeuvre can also be very useful to modify the v_∞ when a number of repeated

[2] A *resonant* leg is the one in which departure and arrival planets are the same. The second consecutive swing-by of the planet is also called resonant.

swing-bys of the same body are used (for example Venus). Automatic design of local minimum Δv manoeuvres between gravitating bodies were included in the systematic search performed by STOUR, thus not changing substantially the way the software operates. At the same time, the new algorithm opens up a whole new realm of possibilities because the inclusion of the manoeuvres creates many more trajectories and allows for greater flexibility in mission design [71]. The inclusion of the Δv manoeuvres in the process lead to a huge number of trajectories that STOUR had to analyse, for each one of the possible planetary sequences. In addition, the total number of sequences could be also very high. This translated in long computational time required by STOUR. To counteract this problem, different methods were investigated, mainly to reduce the number of planetary sequences to be investigated by means of STOUR. The first method was applied to the search for a tour of the Galilean satellites of Jupiter for the Europa orbiter [73]. It was based on the Tisserand plane [37]: this is an analytical plot of the period vs. perijove of the orbits around Jupiter coplanar with the Galilean satellites. Contours of constant v_∞ for each satellite, assuming circular and coplanar orbits can be drawn. Since in each swing-by the energy is conserved with respect to the satellite, the manoeuvre corresponds to a movement along one of the contours. Therefore, given an initial orbit of the spacecraft, it is possible to visually identify on which satellites a gravity assist is possible, and which is the resulting orbit (see Appendix E for details). This technique, which is essentially what was proposed in [36], does not take into account the phasing problem: in other words, it is assumed that a swing-by is possible whenever an orbital intersection exist. Certainly, when the full problem with planetary ephemeris is taken into account, the same swing-bys happen rarely or never. In Ref. [37], the authors also proposed to compute the minimum flight time for a given path in the Tisserand plane, by studying, for each leg, all the possible transfer arcs between the circular orbits of two satellites. This analytic technique, combined with the ability of the mission analyst at tracing promising paths on the Tisserand plane, allowed the identification of a limited number of planetary sequences, which could then be assessed by STOUR in a relatively short time.

Another method to select promising sequences of swing-bys for STOUR is what Petropoulos et al. [74] proposed in 2000. The method made use of what the authors called gravity assist plots (GAP): essentially each plot shows the maximum reachable aphelion for a given sequence of planetary swing-bys in the solar system, as a function of the launch excess velocity from Earth. The trajectories, as in Ref. [37], are fully ballistic, and based on the assumption of circular and coplanar orbits of the planets. The phasing problem was also neglected and the planets were assumed to be at the required place for the swing-by. The outgoing velocity from a swing-by was computed assuming the maximum deflection angle. The GAP analysis produced a rank-ordered list of sequences that reach Jupiter. Other non-classical sequences were obtained through variations of the standard sequences. The sequences were then examined over a given launch period with STOUR, leading to several promising trajectories.

In 2004, Petropoulos and Longuski presented an extension of STOUR, for the design of low-thrust trajectories [75], named STOUR-LTGA. In STOUR-LTGA,

low-thrust arcs were modelled with a shape-based method [76]: a particular shape of the powered arc was used, such that the thrust profile could be determined (the authors opted for an exponential sinusoid); the parameters of the shape were then adjusted to fulfil the boundary conditions. This solution was not optimal but very fast to obtain. The choice of these parameters, for each thrust arc, was included in the systematic scan of the STOUR core. The approach was proven to be successful to quickly individuate a wide choice of preliminary solutions. The tool was applied to a range of test cases, in addition to be used for the GTOC1 problem [67].

OTHER TOOLS

STOUR was not the only tool based on the two-level approach. A number of other methods were developed in the past years for space trajectory design.

Pessina et al. [45] proposed a preliminary tool for automatic search of interplanetary trajectories, combining gravity assists and aero-gravity assists (AGA). The main idea was that, during a low-altitude swing-by, a lifting body performs a flight through the atmosphere of a suitable planet; exploiting the aerodynamic forces, it augments the total deviation angle δ of the $\mathbf{v}_\infty$ vector, with respect to a manoeuvre assisted only by gravity. In addition, due to the aerodynamic drag that acts on the probe during the whole AGA trajectory, the modulus of the outgoing $\mathbf{v}_\infty$ is no longer the same as the incoming one, as it is for the ballistic manoeuvre. We will not focus on the different AGA models, as aero-gravity assists are not included in this thesis, but rather on the search for all possible multiple swing-by trajectories. The tool PAMSIT (preliminary analysis of multiple swing-bys interplanetary trajectories) used a simplified model of the Solar System with circular coplanar orbits of the planets: assuming the spacecraft motion to be in the same plane of the planets, it looked for quasi-ballistic solutions, systematically spanning all the solution space. PAMSIT performed the analysis of the solution space at two different levels of model complexity: what was called A-type study, which was an energy-based feasibility study (no phasing), and a B-type study, where phasing of the planets was taken into account. In the A-type study, given a specific target, all possible permutations of intermediate planetary flybys (GAM or AGA) were analysed, ignoring the phasing problem. Only trajectories that remained hyperbolic (relative to the planet) for the whole flybys were considered. The launch excess velocity and all the trajectory parameters were discretised. Once determined the orbital parameters of one phase between two consecutive planetary encounters, all the possible transfers were investigated and the different times of flight were calculated (assuming less than one complete revolution), in a similar fashion to what is proposed in [73]. Resonant transfers were also considered. For each different planetary sequence, among all feasible trajectories, the one with the lowest time of flight was saved. Despite the lack of the phasing information, the A-type study provided the user with a lower bound for the time of flight. Furthermore, it found which sequences were feasible using only energy-based considerations, thus reducing the computational time of the subsequent analysis, which introduced the phasing model, inasmuch as unfeasible solutions could be now disregarded. In the B-type study, the systematic approach was preserved, but the mean motion of the planets on the circularised coplanar orbits was introduced. By varying the launch

date, the position of the spacecraft at the swing-by dates was constrained to match the position of the planets, within predefined tolerances. A finite number of bound orbits could be performed before each planetary encounter. Sets of interesting solutions could be selected, choosing suitable merit functions made by a combination of time of flight and Δv. PAMSIT did not make use of any optimisation technique: it relied on a simplified model of the Solar System and discretisation of the parameters that allowed using a systematic search in a reasonable computational time. The two-level approach allowed mainly searching for a set of energetically feasible planetary sequences, and then introduced the phasing problem. Since the search in the B-type study could be much more demanding than the A-type, the latter allowed reducing the candidates to be assessed by the former. It is also interesting to note that, despite the use of planar orbits, together with the tolerance in the planet positions, the solutions found by PAMSIT could be locally re-optimised with full ephemerides, and similar solutions were found.

Vasile et al. [69] also investigated low-thrust trajectories to Europa, the moon of Jupiter, including resonant fly-bys in the Jovian system. The optimal swing-by sequences and the first guess solution for the optimal control problem (which was solved with a direct method [77]) was found by using impulsive-manoeuvre reduced models of increasing complexity. They were conceptually similar to STOUR, and made use of a systematic search. The design of the first guess was a two-step process: the first step identified a large number of sequences and a possible interval of launch dates, using a very simple model. The second step computed an accurate multi-impulsive trajectory. The outcome of this second step was then used for the low-thrust optimisation.

The method used at DEIMOS [68] for the GTOC1 problem made use of a hybrid optimisation process in order to reach optimal solutions, based on a three-step strategy: first, a systematic scan of the global solution space, followed by a manual selection of the best-suited options which were finally optimised by means of a local low-thrust optimiser. The first step of the strategy used ballistic Lambert arcs and systematic scan combined with non linear programming optimisation. Discrete manoeuvres were included only at the arrival branch or at the departure branch of the swing-by hyperbolas: this last assumption allowed having only purely ballistic arcs, both for deep space flight and swing-by phases, as the manoeuvre was applied at the interface between the two phases, patching the two conic sections. Furthermore, the degree of freedom due to the positioning of the DSM was removed. Note that, although the manoeuvre does not occur at the pericentre of the swing-by, this model is algorithmically analogous to the powered swing-by model that will be presented in Section 2.3.2. The second step selected the best solutions, among all those obtained at the previous step, based on three criteria: best cost function values; low total required Δv; and short mission duration. In the third and last step, the few selected solutions were refined by means of a local constrained optimiser using a direct method. Although the complete systematic scan of solutions, followed by the optimisation of the ballistic transfers, ensured that the best optimal cases are not missed in the process, the authors did not provide any automatic method for the selection of the planetary sequence. Within the context of

GTOC1, only three sequences were selected, discarding *a priori* a large number of combinations of swing-bys. Nonetheless, the three selected sequences produced about 47,000 solutions after the first step.

In 2007, Olds et al. [62] investigated the relationship between the performance of DE, applied to the design of MGA trajectories, and its tuning parameters over a set of test cases. They incorporated DE in a software code called mission-direct trajectory optimization program (MDTOP). MDTOP implemented a trajectory model with powered swing-bys and ballistic interplanetary legs. DSMs were allowed along some (user specified) arcs. The tests were performed on several complex interplanetary missions: Cassini, Galileo, a sample return to comet Tempel 1 and a round trip to Mars with orbital rendezvous. Due to the stochastic nature of DE, 1000 runs were performed for each test case. The analyses in the work of Olds et al. led to the identification of a particular tuning of DE that is optimal for the set of tested cases. This approach could be considered the second layer of a two-level approach: in fact, MDTOP assumed that the planetary sequence is given, as well as which legs contain a DSM.

PRUNING THE MGA SOLUTION SPACE: GASP

Pruning, in machine learning and global optimisation, is the technique that reduces the size of a decision tree (or more in general search space) by removing some sections of it (sometimes called *branches*), based on some knowledge or heuristics [78]. A different approach to finding solutions to the optimum MGA transfer problem is that of pruning the solution space, such that the *residual space* (the remaining space after pruning) contains all solutions at an arbitrary distance ε from the global optimum, in the criteria space.

This type of approach can be catalogued as a two-level approach, mainly because it does not address the problem of the planetary sequence. This must be given, and the pruning is performed only on the (real valued) space of the possible trajectories encountering the planets. In other words, the pruning is done only on the continuous part of the problem.

The most important work on this topic was done by Becerra, Myatt, et al. [61, 79]. The authors consider a multi-impulse trajectory model, with no DSMs: the changes in velocity provided by the engine happen during the swing-by manoeuvres (powered swing-bys). A similar model (but extended with DSMs) will be explained in detail in Section 2.3; for the moment, it is enough to say that this model allows the definition of each planet-to-planet leg by means of the departure and arrival epochs only. Therefore, all the transfers connecting two planets in a sequence can be generated by a systematic scan of a two-dimensional grid, independently of all previous and subsequent pairs in the sequence.

Some problem-dependent pruning criteria are used to remove, from each two-dimensional grid, sets of departure and arrival epochs. For example, in the first arc, that starts with the launch, the initial excess velocity cannot be higher than what is provided by the launcher. Other constraints come from matching the incoming and outgoing velocities at a the planetary swing-by: since the swing-by is powered, a pair of incoming and outgoing velocities is accepted if the required corrective impulse is lower than a given threshold and pericentre altitude is above a minimum

limit. Hence, for a given pair of planets, an epoch is pruned out from the grid if no trajectory is arriving to or departing from the planets at that epoch (see Fig. 1.8).

These ideas are implemented in the algorithm GASP (Gravity Assist Space Pruning). The residual search space can be searched using general global optimisation methods. The authors propose to use Differential Evolution, Particle Swarm Optimisation, Multiple Particle Swarm Optimisation or Genetic Algorithm and demonstrate the effectiveness of GASP on several test cases, among which the Cassini sequence EVVEJS.

Myatt, Becerra et al. demonstrated [59] that, if the trajectory model does not contain DSMs and a powered swing-by model is adopted for the gravity assists, then an algorithm with polynomial complexity exists that can prune the solution space efficiently. This property is peculiar to the considered instance of the MGA problem, because each planet-to-planet transfer can be decoupled from the others.

Despite this algorithm results to be extremely successful in pruning the solution space, there are two main limitations.

The first one is that the propelled Δv's are allowed only during swing-by manoeuvres, at the pericentre of the hyperbola. The change in orbital parameters during a powered swing-by manoeuvre is very sensitive to the applied Δv. Since the real manoeuvre is not instantaneous, and thus not confined at the pericentre of the hyperbola, the actual variation in orbital parameters can be quite different than what is predicted with the powered swing-by model. Moreover, a manoeuvre at the pericentre of the swing-by hyperbola poses strict requirements in terms of operations: the spacecraft, within the sphere of influence of the body, must be observed and its orbit accurately predicted and possibly corrected from the ground, within a short time. Also note that, if powered swing-bys are used to model trajectories in which the actual burn is performed in deep space, then super-optimal solutions are likely to be found. This will be shown in Section 2.3.4.

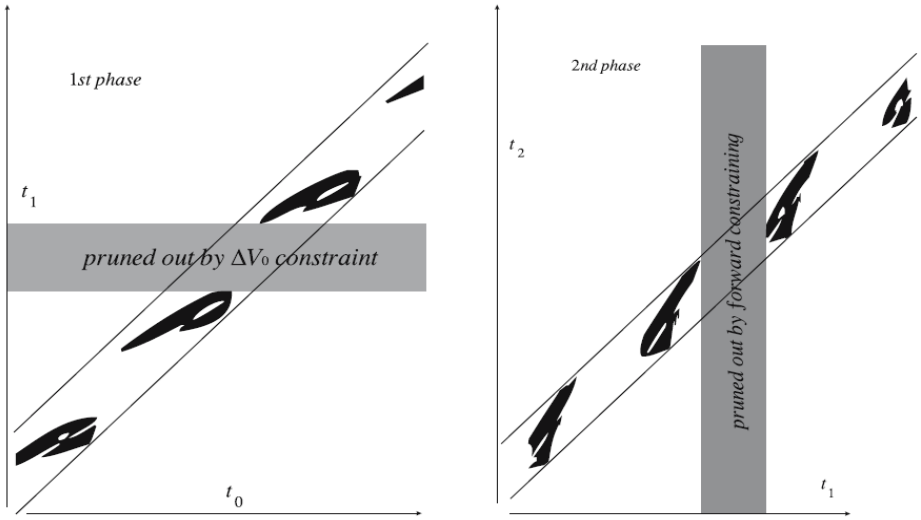


Fig. 1.8. Pruning of the search space using GASP. From [79].

In addition, an MGA model with no DSMs does not allow the design of a number of interesting trajectories. The main examples of these types of transfers are planet-to-planet resonant swing-bys. In resonant transfers a DSM is used to change the relative velocity with respect to the swing-by planet, and it can be shown that a small manoeuvre can produce a consistent change in asymptotic velocity. There are other (non-resonant) cases in which transfers can only be achieved through a DSM, and some will be presented in this thesis (see for example Section 4.2.2).

Finally, the introduction of DSMs offers the further advantage of providing a reasonable approximation of MGA trajectories with low-thrust arcs [69], thus allowing the generation of first guess solutions also for that kind of trajectories. The idea is that the change in velocity provided by the impulsive manoeuvres is spread along propelled arcs, when converting to low-thrust propulsion. This operation is usually performed solving an optimal control problem. If no DSMs are considered, and thus the arc is purely ballistic, the only impulsive manoeuvres are at the beginning and at the end of an arc. On the other hand, if one or more DSMs are inserted, part of the original change in velocity is taken by the DSM itself. Therefore, there will be at least three impulsive manoeuvres along the arc, in average with a smaller Δv: this trajectory, then is more similar to a low-thrust solution.

The second issue is connected to the first, and related to the systematic scan: as mentioned above, each two-dimensional sub-space is explored with a simple grid sampling. This approach is suitable as long as the dimensionality of the grids remains limited, and this is the case in GASP, due to the particular trajectory model. Unfortunately, if DSMs are inserted along a planet-to-planet transfer leg and an unpowered gravity assist manoeuvre is considered [80], the number of degrees of freedom increases significantly, hence the grid sampling approach becomes problematic due to the higher dimensionality of each sub-problem; if a coarse grid and an aggressive pruning are used, many optimal solutions may go lost; on the other hand, if a sufficiently fine grid is used, the computational time becomes unacceptable even for a limited number of planetary swing-bys. As it will be explained in the remainder of this dissertation (see Section 2.2), this is due to a dependency problem peculiar to this particular way of modelling MGA trajectories.

If the problem of MGA with DSMs (MGADSM) could be pruned in polynomial time with a small exponent, an efficient branch and prune algorithm could be used as in GASP. Hence, an efficient deterministic solution process would have to make use of additional information (with respect to the simple ballistic case) to cut down the number of possible alternatives.

REMARKS

The main issue with two-level approaches is to assess the optimality of a given planetary sequence without an exhaustive search for all possible trajectories associated to that sequence. Unfortunately, finding an optimal trajectory is a very difficult global optimisation problem in itself. This, combined with the fact that usually there exist a very high number of sequences for a given transfer problem, requires a considerable computational effort. The computational cost can be reduced by discarding non-promising sequences. However, if the low-fidelity model

is not accurate enough, either some good sequences are discarded, or many of the retained ones can result to be not good.

1.4.2 Integrated Approaches

As opposed to the two-level approaches, integrated approaches define a mixed integer-continuous optimisation problem, which tackles both the search of the sequence and the optimisation of the trajectory, using a single model, at the same time [41]. This kind of problems is known in the literature as a hybrid optimisation problem [81].

HYBRID OPTIMISATION

The theory of hybrid optimisation is at very early stage, but in rapid development. The challenges in solving these types of problems are mainly due to the intrinsic combinatorial complexity, in addition to the nonlinearity of the continuous, multi-phase optimal control problem. For example, von Stryk and Glocker [82] proposed a branch and bound approach for mixed integer optimal control problems. They consider a general framework in which a binary control is added to the classic continuous control. Essentially the branch and bound is applied to search the entire discrete solution space: by partially relaxing the binary variables, the inner nodes of the tree define an optimal control problem, whose solution provides a lower bound on the performance index for all the nodes of the sub-tree. If the lower bound for a given sub-tree is greater than the current global upper bound, then the entire sub-tree does not need to be searched. The authors show the effectiveness of the method by applying it to the solution of the Motorised Travelling Salesman Problem (MTSP), in which the discrete part is represented by the sequence of the cities to visit, while the continuous part is the control of the car driving from one city to the other.

A current practice to solve hybrid optimal control problems is to use two nested loops: an outer loop problem solver which handles the discrete dynamics, and finds a solution in terms of categorical variables, and an inner loop problem solver that performs the optimisation of the corresponding real-valued problem. Since a solution of the inner problem is required for each function evaluation of the outer one, the outer loop has to find the optimal value of the categorical variables in fewer evaluations than a systematic enumeration would require.

The automatic design of a trajectory with discrete events was recently formulated by Ross et al. as a Hybrid Optimal Control Problem [81]. Wall and Conway [83] proposed an integrated approach with hybrid optimisation, based on genetic algorithms, using the two-loop approach. Here the outer loop generates a sequence of asteroids to be visited, while the inner loop optimises the corresponding trajectory. For the outer loop, they used two solvers: a branch-and-bound and a GA. Since the authors consider a low-thrust spacecraft, in general direct methods are common to solve the inner loop optimal control problem, due to for their robustness. The authors instead decided to use again GA, applied to a shape based method for low thrust trajectory design. These combinations are tested on three

cases of increasing complexity. All of them involve the optimisation of a multiple asteroid rendezvous trajectory.

In 2003, Vasile [52, 66] proposed a mixed approach to global optimisation, combining a deterministic branching technique and a particular implementation of evolution programming (EP). The former, being predictable, has the task of guarantee that the algorithm can find the global optimum in a predictable amount of time. The latter part, instead, being stochastic, cannot be proven to converge to a global solution within fixed time, but on the other hand it has the task of obtaining a lower bound information, to select promising branches and to prune non-promising ones. The resulting algorithm named Evolutionary Predictive Interval Computation (EPIC) was shown to be effective for solving some typical problems in space mission design. The test cases considered in that work were: a ballistic transfer from Earth to Mars; a low-thrust transfer between the same planets, in which the solution of the optimal control problem is substituted by assigning a given shape for the thrust control profile throughout the transfer; and finally a MGA from Earth to Pluto, in which two swing-by manoeuvres were inserted in the trajectory. The choice of the planets that offer the gravity assist was left free to be found by EPIC, exploiting its ability to deal with mixed integer-continuous problems.

A MGA trajectory optimisation tool was also developed by Vasile and De Pascale, and named IMAGO [41] and implemented in MATLAB®. It combines a trajectory model with MGA and DSMs together with EPIC. The tool is tested on a number of cases, including a near-Earth orbit interception mission and a Rosetta-like mission. The tool, exploiting EPIC's ability to optimise categorical variables, is also applied to a MGA transfer to Jupiter, with free planetary sequence, i.e. EPIC is free to choose, from a selected set, which triplet of planets to use for the gravity assists.

In 2009, Chilan and Conway [84] developed an approach to the automated planning of a sequence of impulses, thrust arcs and coast arcs for a trajectory with given boundary conditions. Finding the optimal sequence of events along a trajectory is seen as a planning and scheduling problem. The plan was encoded into a vector of discrete, or categorical, variables, each of which represents an event. Each feasible sequence of events was then associated to a cost resulting from the solution of a hybrid optimal control problem. The authors tackled this problem using two nested loops: the outer one browses through the values of the categorical variables, whilst the inner one solves the optimal control problem. Chilan and Conway proposed to use genetic algorithms for the outer loop. The possible events forming a plan were: coast arcs, impulses, boundary-free thrust arcs, boundary-specified thrust arcs, Lambert arcs. A general combination of these events may lead to unfeasible sequences: for example, a transfer to a planet cannot end with a boundary-free thrust arc. By simple considerations on the nature of the events, they found a way to encode, using a binary vector, sequences of events of variable length (up to a maximum number), which are always feasible. The optimal control problem is transformed into an NLP problem, with a modular scheme for the automatic construction of the mesh depending on the scheduling of the events. Two methods, both based on GA, are proposed for determining the first guess that initialises the NLP solver.

Pisarevsky and Gurfil [63] proposed an application of memetic algorithms (MA) to the optimisation of MGA trajectories to reach Jupiter and then a highly inclined orbit. MAs combine a population-based search with separate individual learning, i.e., local refinement operators. The authors designed optimal MGA trajectories to reach Jupiter with high relative velocity, low Δv and transfer time. No particular method for selecting sequences is presented. Local search was then applied to most promising solutions to solve the swing-by at Jupiter and finally achieve the high inclination orbit.

REMARKS

The main difficulty with integrated approaches is that a variation of even a single celestial body in the sequence corresponds to a substantially different set of trajectories. Therefore, if the solution of the hybrid optimisation problem is represented with a single vector, a variation even of a single integer number in the sequence corresponds to a substantially different set of trajectories. Furthermore, a variation of the length of the sequence implies varying the number of legs of the trajectory, and thus the total length of the solution vector.

1.5 Study Objectives

Finding the global optimum to a generic MGA optimisation problem is a challenging problem. As mentioned before, the problem is even more complex if DSMs are considered. The objective of this study is to investigate new methods for the automatic design of MGA trajectories with DSMs. These new methods should lead to tools that work with minimum experience and intervention of the mission analyst.

Both a two-level approach and an integrated approach will be covered in this thesis.

1.5.1 Incremental Pruning

First, a two-level approach for MGA trajectory design including DSMs will be investigated. The primary objective is to extend the results obtained with GASP [59, 79]. Different models for MGA trajectories with impulsive DSMs will be defined and it will be shown that each model leads to an algorithm with a different level of complexity.

The idea developed in this thesis is that, no matter the model, the MGA global optimisation problem can be decomposed into simpler sub-problems (of smaller complexity and size), and approached incrementally, adding one planet at the time to the trajectory. At each incremental step, a portion of the search space can be pruned out. In fact, starting from the departure planet and flying to the first swing-by planet, only a limited set of transfers are feasible, and the rest of the search space can be pruned out, according to a number of criteria. The process iterates adding the second planet in the MGA sequence, and continues until the final planet is reached.

The main part of the work will assess how to effectively and efficiently prune the solution space in case DSMs are included in the trajectory description. A first result will be the definition of a set of efficient pruning techniques for this specific problem. The proposed pruning techniques allow a fast solution of problems with a number of gravity assists and DSMs.

It will be shown that, because of the increased dimensionality introduced by DSMs, grid search ceases to be efficient and can be replaced with non-exhaustive heuristics. The heuristics developed in this thesis aim at finding the set of solutions fulfilling the pruning criteria for each sub-problem.

Appropriate pruning criteria will be developed, such that not only will the incremental approach find the minimum-Δv transfers, but also take into account other mission requirements.

The effectiveness of the pruning techniques will be demonstrated on a benchmark of realistic mission design problems. It will be shown that the exploration of the residual space (i.e. the non-pruned space) provides better quality results, in less computation time, than the solution of the problem as a whole.

The search space reduction obtained with the developed pruning techniques allows a reliable identification of: launch windows, mission options over a wide range of launch dates, mission cost in terms of Δv and epoch of the manoeuvres for a given sequence.

Finally, the tool will be proven to be able to work on a wide variety of MGA transfer problems: from interplanetary trajectories (possibly including resonant legs), to a tour of the Jovian moons, to sequences of resonant swing-bys to change the orbital parameters. Test cases include the design of trajectories for the BepiColombo and Laplace missions.

PLANETARY SEQUENCE GENERATION

Like all the two-level approaches, the generation of the planetary sequence is performed independently and in advance, by means of a reduced trajectory model. Therefore, an additional tool is necessary for finding a set of planetary sequences to be investigated with the incremental pruning approach. The idea is to have a conservative approach: it is preferred to have a higher number of candidate sequences, rather than have few or just one, with the risk of eliminating good ones.

The generation of candidate sequences starts from a list of available planets to swing-by. All the possible sequences are examined incrementally, starting from direct transfer, adding one and eventually more swing-bys. The list is pruned out first by considering some constraints, like the maximum number of swing-bys and the maximum number of resonant swing-bys. Then, their feasibility is assessed by means of a simplified model, based on energetic considerations, in a similar flavour to what proposed by Petropoulos at al. [74]. Unfortunately, due to the choice of having maximum deflection at each swing-by, this framework does not provide any estimation of the relative velocity v_∞ at the final planetary encounter. The value of the v_∞ can be quite important while trading off different solutions: in fact, it could be used to rank all the feasible sequences, and select the most promising ones, for the subsequent optimisation.

Therefore, the approach will be extended to take into account requirements on the arrival velocity with respect to the planet (v_∞). In most transfers, the v_∞ has to be minimised, in order to favour an orbit insertion around the target planet with low Δv. In some other cases, instead, a specified value of relative velocity needs to be achieved, as it is required by the following part of the mission.

1.5.2 Ant System Trajectory Planning

The objective of this part of the research is to develop an approach to the automated planning and scheduling of MGA trajectories.

In the literature, *planning* is the problem of defining a sequence of activities (or actions, or events) to reach a goal. *Scheduling* is the one of finding the best temporal allocation of those activities. A feasible and optimal solution satisfies a set of constraints and minimises a given objective function.

An MGA trajectory can be seen as a scheduled sequence of events (e.g. launch, deep space flight, DSM, swing-by, planetary capture) characterised by a set of integer variables, identifying the type of event, and a set of real variables, identifying the time and characteristics of the event. From this point of view, the definition of an MGA trajectory becomes planning and scheduling the events and its optimal design translates into the solution of an optimal planning and scheduling problem.

Since planets are moving and can be revisited (e.g. in resonant swing-bys), the MGA planning problem can be seen as a variant of the dynamic vehicle routing problem [85] in which each node of the network can be revisited, the customers are moving and the cost of each leg of the itinerary depends on the route followed by the vehicle up to that leg. Similar to other NP-hard problems, the number of possible paths, for an MGA transfer, grows exponentially with the number of planetary encounters.

Here, the automated MGA trajectory planning problem will be tackled with an integrated approach in which the trajectory model is an integral part of the solution process. In particular, the model contains a scheduler that transforms a trajectory, with the associated mixed integer-continuous variables, into a finite space of scheduled discrete events: the plan space.

An Ant System (AS, [86]) algorithm is then used to look for optimal plans. The main idea is to dispatch a group of virtual ants into the plan space. Each ant will choose to follow a particular plan (or path in the plan space) according to a probability distribution. The probability associated to a particular plan is proportional to the quality of the corresponding set of previously explored MGA trajectories. This process is analogous to the deposition of the pheromone in standard Ant Colony Optimization (ACO, [87]). The next sub-section is a literature review of works that used ACO to solve planning and scheduling problems or routing problems. However, unlike in standard routing problems, here the nodes are moving and the cost of the transfer from one node to another depends on the previous history of allocated actions. Therefore, it is possible to associate a pheromone level to a complete path but not to a portion of it.

In order to solve this dependency problem, the proposed algorithm does not use standard pheromone deposition and evaporation heuristics but employs taboo lists, to guide the decision of the ants at each planet (or node). An external archive maintains a list of complete plans, the *feasible list*, and is used to build a statistical model of the feasible set within the search space. Dual to the feasible list, the taboo lists are a surrogate representation of all the infeasible choices (i.e. choices that do not lead to completing a plan). In this way, the ants avoid the re-exploration of paths that resulted to be infeasible, and direct the search towards other more promising areas of the plan space.

The effectiveness of this approach will be proven through the solution of a number of relevant MGA trajectory design problems: an Earth to Mercury transfer, similar to the one of the BepiColombo mission, and an Earth to Saturn transfer, inspired to the Cassini mission.

1.5.3 Planning and Scheduling with ACO

In this thesis, it is proposed to solve the path planning problem, associated to the design of MGA trajectories, with a modified Ant Colony Optimization (ACO) [87] algorithm. ACO was originally created to solve the travelling salesman problem (TSP) [86], and later successfully applied to a number of other discrete optimisation problems.

In the literature, some ACO-derived meta-heuristics exist for the specific solution of different scheduling problems.

For example, the resource-constrained project scheduling problem (RCPSP), aims at finding the optimal sequence of scheduled activities of a project, such that the time difference between the start and finish of the schedule is minimized. Constraints are posed on the ordering of the activities and on the resource requirements per time unit. Merkle et al. [88] proposed to apply ACO to decide which activity, from the set of eligible ones, should be scheduled next. Both pheromone information and heuristics were used for selecting the activities.

The open shop scheduling (OSS) problem was also tackled through ACO. A set of machines have to perform a given number of operations taken from a set. A machine can perform only certain operations. Jobs are sets of operations which define additional constraints, as operations in the same job have to be performed sequentially. A solution of the problem is defined by the sequences of operations on each machine. Each operation has a given processing time, and the objective of the problem is usually to minimise the makespan, while satisfying the constraints. Blum in his work [89] suggested to hybridise Ant Colony Optimisation with Beam Search (BS) for solving the OSS. The idea came by considering that a scheduling problem can be seen as a search in a tree, and thus algorithms specifically developed for this purpose can be used. Beam Search is a (usually deterministic) technique to construct several candidate solutions in parallel, and it uses a lower bound in order to guide the search; in contrast, ACO algorithms explore the search space probabilistically, using past experience through pheromone. Expecting a benefit from combining the two, Blum used the ACO as a basic framework, but replaced the solution construction mechanism by a probabilistic BS.

Recently, some authors have also hybridised Ant Systems (AS) with taboo search for applications such as the Job-Shop Scheduling problems [90, 91] or the data clustering problem [92]. The algorithms proposed in the literature use taboo search to improve locally the solutions or to avoid re-visiting nodes in a tree. The modified ACO algorithm in this paper, instead, employs taboo lists to discriminate between feasible and infeasible paths. In fact, because of the dependency problem, at every node, feasible and infeasible directions are undistinguishable without the memory of the past moves of the ants.

1.6 Document Structure

This thesis is organised as follows. Chapter 2 will describe two different MGA trajectory models including impulsive DSMs. One model makes use of non-powered swing-bys and one DSM per interplanetary leg, while the second can have multiple DSMs per leg and makes use of powered swing-bys. The chapter contains an analysis of the influence of each model on the complexity of the solution algorithm.

In Chapter 3, the two-level approach will be presented. The chapter is mainly focused on the development of the incremental pruning technique but contains also a description of the method to generate sequences of gravity assists. At the end of the chapter, some comparative tests between the pruning approach and other global optimisation methods (both deterministic and stochastic) will be presented.

Chapter 4 will show the application of the proposed sequence generator and incremental pruning techniques to real mission test cases. This chapter will also include extended analyses of the results, including an investigation of the different families of solutions that the incremental approach is able to find.

The novel integrated approach based on ACO will be described in Chapter 5. This chapter will include the particular trajectory model specially developed for this approach, the search technique, the application to different interplanetary transfer problems and a comparison with other stochastic methods.

This thesis ends on Chapter 6 with some final remarks and suggestions for future investigations.

There are five appendices in this thesis. Appendix A describes some code implementation details. Appendix B shows a trajectory composition approach based on the definition of basic *building blocks*. This approach allows assembling trajectories that combine building blocks taken from the two models, low-thrust arcs and any other type of event. Appendix C presents a transformation of the search space that allows to transform a set of disconnected or overlapping boxes into the unitary hyper-cube. Two test cases are also presented. Appendix D presents a testing procedure for global optimisation algorithms. Finally, Appendix E gives an overview of the Tisserand plane.

2

MGA TRAJECTORY MODELS

This chapter introduces two different patched-conics MGA trajectory models: the velocity formulation and the position formulation. The former describes each segment of the trajectory starting from its initial velocity, uses an unpowered swing-by and allows one DSM per leg. The latter describes each segment through the position at the bounds, uses a powered swing-by and more DSMs per leg. It will be shown how each model affects the complexity of the MGA design problem.

2.1 The Patched-Conic Approximation

Throughout this work, the trajectories will be modelled using the *patched-conic* approximation [93]. The fundamental assumption underneath this approximation is that the motion of a point, with negligible mass, subject to the gravitational action of multiple bodies, can be described as the union of a number of individual conic arcs along which only one gravitational action is active and the others are null. Each conic arc is the closed form solution of the Newton's law of universal gravitation:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\hat{\mathbf{r}} \,. \tag{2.1}$$

with $\mathbf{r}$ the position vector of the point mass with respect to the active gravitational body, whose gravitational constant is μ, and $\hat{\mathbf{r}} = \mathbf{r}/r$. The dot $\dot{\square}$ denotes differentiation with respect to time. Eq. (2.1) implicitly assumes that any other force (e.g. solar pressure) is negligible. The motion following Eq. (2.1) is called *Keplerian*, and it can be shown that the trajectory is a conic section.

This assumption applies well to the case of MGA trajectories. A spacecraft moves in the deep space under the attraction of the main attractor (the Sun, for trajectories in the Solar System), and performs a swing-by every time it enters the sphere of influence of a gravitational body. The *sphere of influence* (or Hill's sphere) is defined as the region around a planet within which the gravitational attraction of the planet is prevailing with respect to those of all the other bodies.

Therefore, inside the sphere of influence, the motion can be approximated by considering the gravity of the planet only. Note that, in order for the patched-conic assumption to hold true, the transition through the Hill's sphere should be fast enough to be approximated with a hyperbolic arc.

A limit case of the patched-conic framework is obtained considering that the spheres of influence of the bodies have infinitesimal radius and the swing-by is instantaneous. In fact, this approximation is not far from the reality, when considering planetary systems (like the Solar System) in which the mass of the main attractor is several orders of magnitude bigger than the masses of the other bodies. In this approximation, the sphere of influence each body degenerates into a point, therefore the spacecraft actually targets the planet by matching its position at a given time. This also implies that the point in which the spacecraft enters the sphere of influence of the body is not defined: this leaves an additional degree of freedom to be fixed in the swing-by phase, as it will be shown later. This limit case of the patched-conic approximation is sometimes referred to as *linked conics*.

Following this approximation, the complete MGA trajectory can be decomposed in different parts.

An *interplanetary leg*[1] connects one celestial body to the next one. Each interplanetary leg is made of one or several *arcs*, in which the spacecraft moves under the attraction of the main attractor (the Sun in the solar system, for example) and possibly additional forces due to perturbations or its own thrust.

Arcs are characterised by having a non-null duration, as opposed to instantaneous events, that connect two arcs matching the spacecraft state before and after according to certain rules. Instantaneous events can happen at planets, like launch, swing-by or capture, or in deep space, like a DSM.

In the remainder of this section, the various parts composing an MGA trajectory are presented in detail.

2.1.1 Arcs

An *arc* is a non-instantaneous part of a leg along which the states of the spacecraft (position and velocity) are continuous functions of time. Arcs can be further classified in two categories: coast arcs and low-thrust arcs.

In general, along a *coast arc*, a spacecraft is subject only to natural forces, e.g. gravity, solar pressure, etc., and not to any control action coming from the use of a propulsion system. In the remainder of this thesis, a coast arc is further restricted only to purely Keplerian motion, i.e. the spacecraft is subject to the attraction of one single gravitational body and no other forces are influencing its dynamics. In this case, therefore, a coast arc is always a part of a conic section.

Along a *low-thrust arc*, instead, a spacecraft is subject to a controllable low-thrust action coming from the use of a propulsion system. If other natural forces are acting on the spacecraft along a coast arc, the coast arc is called a *perturbed coast*

[1] We will refer to interplanetary legs, with some abuse of terminology, not only for Sun-centred trajectories, but also for planet-centred trajectories. In the latter case, the bodies are the moons of the planet.

arc, and the additional forces are called perturbations. Of course it is possible to have a *perturbed low-thrust arc*, in which there is both thrust and perturbations.

The classification of the arcs used so far takes into account the dynamics along the arc itself, according to the forces acting on the spacecraft. There is another way of classifying the arcs, which is interesting for the purposes of this work, and it considers the way an arc is computed.

A coast arc can be computed mainly in two ways: by propagation or by solving a Lambert's problem, as it will be explained in the following two sub-sections.

PROPAGATION

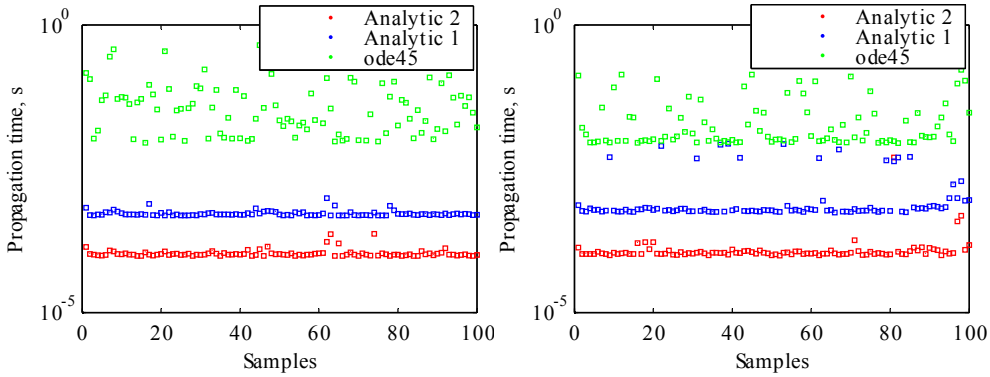
Propagation refers to finding the state of the spacecraft at a given time, by solving the equations of motions, given the initial states (position and velocity) of the spacecraft at some earlier initial time. If the propagation applies to the coast arc, then the two-body equations of motion are integrated. The idea of propagation can be applied to any other type of arc, provided that the equations of motions and the external forces (controls and perturbations) are known.

The propagation can be done in several different ways. Since the equations of motion are a set of differential equations, in general it is possible to integrate this system numerically. For the very special case of non-perturbed Keplerian orbit, analytic solutions exist to find the final states after a given amount of time. Usually these formulations do not require any numerical integration, but only finding the root of a non-linear function. Due to the existence of ways to find good approximations of it, few iterations of a Newton-Raphson loop are usually sufficient to solve the problem.

The analytic solution to the equations of motion offers increased precision in the final states and reduced computational time with respect to the numeric propagation (Fig. 2.1). For this work, an analytic propagation [94] based on universal variables [95] was used: universal variables have the additional advantage that can be used for elliptic, parabolic and hyperbolic orbits with no modification.

LAMBERT ARC

A Lambert arc is a coast arc which is obtained by solving a Lambert's problem [95]. The Lambert's problem is the problem of finding the orbital parameters of a conic arc connecting two points $\mathbf{r}_1, \mathbf{r}_2$ in space in a given time Δt and number of full revolutions (see Fig. 2.2). Once the orbital parameters have been determined, we are usually interested in finding the velocity vectors $\mathbf{v}_1, \mathbf{v}_2$ at the boundaries of the arc. This work makes use of a multi-revolution Lambert solver [96] implemented in C.



(a) Elliptic orbits (b) Hyperbolic orbits

Fig. 2.1. Propagation time for 100 random initial states for elliptic orbits (a) and hyperbolic orbits (b). The three colours refer to numerical propagation (MATLAB® *ode45*), analytic propagation with time equation (Analytic 1) and with universal variables (Analytic 2). All codes implemented in MATLAB®, running on Pentium 3 GHz.

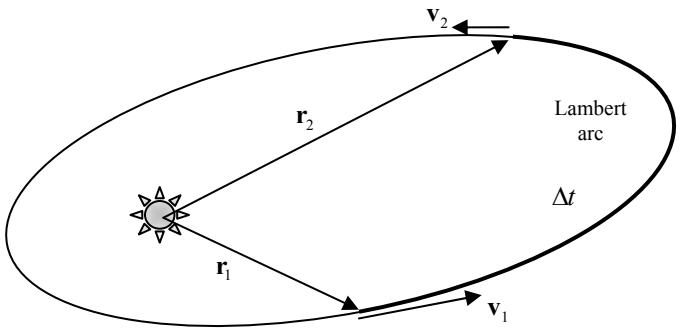


Fig. 2.2. Lambert's problem.

2.1.2 Instantaneous Events

Two arcs can be connected one after the other: in this case, the state of the spacecraft must be continuous across the interface. The typical example is when the spacecraft is coasting in deep space and, at some point, it switches on its ion engine. This situation can be modelled by a coast arc followed by a low-thrust arc. Although there is a discontinuity in the forces applied to the spacecraft, its velocity and position are continuous.

It is obvious that the position of the spacecraft as a function of time must be continuous along the trajectory. This is in theory the case for the velocity, as well. There are some events, though, that can change the velocity of the spacecraft considerably, in a relatively short time. These events are, for example, swing-bys, manoeuvres with a chemical, high-thrust engine, launch, rendezvous, orbit injection and so on. A common approximation, then, is to consider these events as instantaneous. Thus, *instantaneous events* change the velocity of the spacecraft in no time without affecting its position.

Instantaneous events are used to connect two consecutive arcs, which do not have the same velocity at the matching interface.

The following instantaneous events will be considered: deep space manoeuvres, which occur in deep space, and require a thrust, and swing-bys, which exploit the gravity of a body.

DEEP SPACE MANOEUVRE

A *deep space manoeuvre* (DSM) is a change in the velocity vector of the spacecraft obtained by switching on the spacecraft engine. If the thrust is high enough, then a short time is sufficient to produce the necessary change in velocity. Thus, a DSM can be modelled as an instantaneous change in velocity, or $\Delta\mathbf{v}$:

$$\Delta\mathbf{v} = \mathbf{v}_2 - \mathbf{v}_1,$$

where $\mathbf{v}_1, \mathbf{v}_2$ are the heliocentric velocity vectors before and after the manoeuvre respectively, and the modulus of the velocity change is $\Delta v = \left|\mathbf{v}_2 - \mathbf{v}_1\right|$. This latter value is very important, as is directly related with the propellant needed to perform the manoeuvre. In particular, from the well-known Tsiolkovsky rocket equation [97]:

$$m_2 = m_1 e^{-\Delta v / v_e}$$

where m_1 is the mass of the spacecraft before the manoeuvre, m_2 is its mass after the manoeuvre, and v_e is the exhaust velocity of the propellant mass. Therefore, minimising the change in velocity, or Δv, is equivalent to minimising the propellant consumption of the spacecraft.

SWING-BY

A *swing-by*, also called *gravity assist* or *hyperbolic passage*, connects two consecutive interplanetary legs of the trajectory, matching at a given planet [93]. This means that the final point of the first leg coincides with the initial point of the second leg, and both points coincide with the planet position at that time. In fact, the swing-by is also considered to be instantaneous within the interplanetary trajectory, so that the final time of the incoming leg coincides with the initial time of the outgoing leg. Therefore, the position of the planet does not change during the swing-by.

From the point of view of the interplanetary trajectory, the planet is a point, identified by the centre of mass of the planet itself. During the manoeuvre, gravitational attraction of the main attractor is neglected, and the spacecraft is assumed to be subject to the gravity field of the planet only. Two different types of swing-by models will be presented in the following: an unpowered swing-by, in which no manoeuvre is performed, and a powered swing-by. As it will be demonstrated in this study, each one of the two models has substantial implications both from a computational and operational point of view.

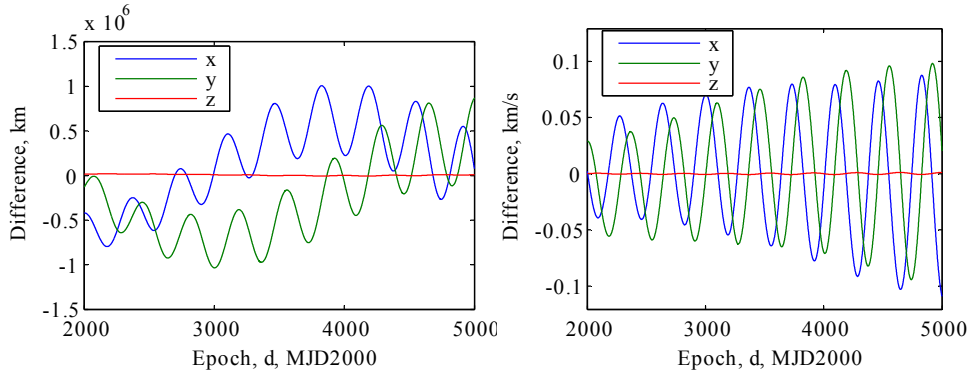
2.1.3 Ephemerides

The positions of the planets, and the other celestial bodies orbiting in a given planetary system, are usually given by the ephemerides. Practically, the ephemerides can be though of as a function which provides position and velocity of a body (or alternatively the Keplerian parameters of the osculating orbit) at a given instant of time, of the type:

$$[\mathbf{r}, \mathbf{v}] = \text{Ephemerides}(b, t) \tag{2.2}$$

where b is the identifier of the body, in a given reference frame, and t is the epoch. The function (2.2) should be defined not only for all the bodies of interest, but also for the whole timespan of the mission.

Since the ephemerides are tables based on real observations, interpolation methods are usually used internally to determine the function at times between observations, and extrapolation is used to determine the function at future times, where observations are still not available. This approach is adopted by the widely used JPL NAIF-SPICE ephemerides [98]. Another approach is possible, instead: an approximation of the orbital parameters can be analytically described in time as polynomials. The evaluation of a polynomial at a given time is many times faster than any interpolation, therefore this second approach provides a computationally cheap version of the ephemerides, at the cost of the precision of the body data. Fig. 2.3 shows the difference for the three components of position and velocity for the Earth, between the JPL ephemerides and the adopted analytical ones. For preliminary mission design, the differences are very small and do not affect the results in any way. Fig. 2.4 on the other hand shows that the analytic ephemeris are more than three times faster than the JPL ones.



(a) Position (b) Velocity

Fig. 2.3. Difference in position (a) and velocity (b) between the analytic ephemerides and JPL NAIF-SPICE "de405" kernel. The body is the Earth, and the range of dates spans from June 2005 to September 2013.

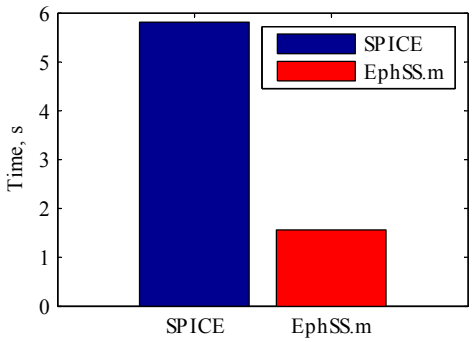


Fig. 2.4. Time for 10,000 calls of SPICE and analytic ephemerides ("EphSS"). The analytic ephemerides are more than 3 times faster.

For this work, the analytic ephemerides were used for the comparative test cases presented in Chapter 3. JPL SPICE ephemerides were used instead for the applicative test cases in Chapter 4, as these ephemerides were in use at ESA-ESOC, and thus the necessity to use exactly the same data, for the sake of comparing the results.

2.2 Velocity Formulation

This section and the following one present two complete, linked-conic, MGA trajectory models with DSMs.

The first model, named *velocity formulation*, composes a trajectory with a sequence of conic arcs linked together in position, assigning a value to the velocity vector at the beginning of each arc. The corresponding position vector is computed as the end point of the propagation of the previous arc, except for the first one. Each arc is propagated forward in time, therefore the position and velocity at the beginning of each arc depend on the previous ones and the whole sequence has to be computed in a strict temporal order.

Here a particular instance of the velocity formulation will be presented. The transfer leg from one celestial body to another phase is composed of a propagated arc, followed by a Lambert arc. The two arcs are connected by a DSM. A non-powered swing-by is used at each celestial body to connect the incoming leg to the outgoing leg.

2.2.1 Interplanetary Leg

Let us consider the generic interplanetary leg in Fig. 2.5, connecting planet P_1 to planet P_2. The time of flight T of each leg is a parameter of the model, thus it is considered as known. The time t_1 at planet P_1 is also a parameter, or can easily be computed by summing all the previous times of flight to the departure time. Then, the time at P_2 is $t_2 = t_1 + T$.

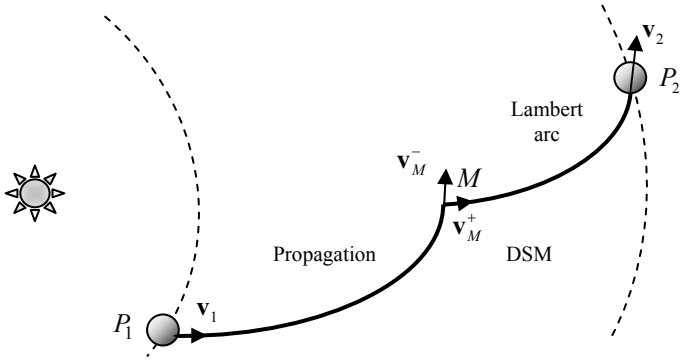


Fig. 2.5. A representation of a leg in the velocity formulation.

We also assume, and this is distinctive of the velocity formulation, that the initial velocity vector $\mathbf{v}_1$ is known. If the considered leg is not the first one in the trajectory, then the initial velocity is the outcome of the preceding swing-by. If instead it is the first leg, then the initial velocity can be either provided, or computed as the result of a launch.

Since the time at planet P_1 is known, its position is evaluated through the ephemeris. The position at the beginning of the leg coincides with the position of P_1, and the full state of the spacecraft (position and velocity) is known just after leaving P_1.

A propagated arc can be computed with these initial conditions, for a given amount of time. The propagation brings the spacecraft to the position *M* and time t_M of the DSM. This time is defined as the fraction α of the time of flight *T*, i.e. $t_M = t_1 + T\alpha$, therefore:

$$\alpha = \frac{t_M - t_1}{T},$$

and the time of the propagation is $T\alpha$.

Note that in the velocity formulation, the position of the DSM is not explicitly defined through a set of parameters in the solution vector (as it happens for the position formulation, discussed in Section 2.3.1): on the contrary, it is the final point of the propagation [41].

The propagation provides the final state of the spacecraft at time t_M, before the DSM, namely position $\mathbf{r}_M$ and velocity $\mathbf{v}_M^-$, where the superscript (-) denotes the fact that the velocity is before the DSM.

The second arc connects the point $\mathbf{r}_M$ at t_M to the position of planet P_2 at time t_2 with a Lambert arc. The duration of the arc is $T(1-\alpha)$, and the solution of the associated Lambert problem gives the velocity vectors at its boundaries, i.e. $\mathbf{v}_M^+$

and $\mathbf{v}_2$ respectively. The DSM magnitude, which represents the cost of the leg, is then computed as

$$\Delta v = \left| \mathbf{v}_M^+ - \mathbf{v}_M^- \right| .$$

The velocity $\mathbf{v}_2$, instead, becomes the incoming velocity to the swing-by at the end of the transfer arc, if existing, or simply the final velocity of the trajectory.

If the initial time and state of the spacecraft are given or computed as the outcome of previous events, the interplanetary leg in the velocity formulation requires two parameters: the time of flight T and the timing of the DSM α.

2.2.2 Unpowered Swing-by

In the velocity formulation, the unpowered swing-by model [93] is used to match two consecutive legs at a planet. We consider the spacecraft to have the same position of planet P. The gravity constant of the planet and its mean radius are respectively μ_P and R_P. Let $\mathbf{v}^-$ be the velocity of the spacecraft, and $\mathbf{v}_P$ the velocity of the planet. In a planetocentric reference frame, the spacecraft is approaching the planet from great distance ($r \cong \infty$) with a velocity, relative to the planet itself, given by:

$$\mathbf{v}_\infty^- = \mathbf{v}^- - \mathbf{v}_P$$

Since the spacecraft approaches from infinity, the orbit is hyperbolic, hence the name of *hyperbolic passage*. We would like to find the outgoing velocity at the end of the swing-by $\mathbf{v}_\infty^+$.

For a hyperbola, the eccentricity $e > 1$, and the semimajor axis a is negative. By using the energy equation at infinity we note that:

$$E = \frac{v_2}{2} - \frac{\mu_P}{r} = \frac{\left(v_\infty^-\right)^2}{2} = \frac{\left(v_\infty^+\right)^2}{2} = -\frac{\mu_P}{2a} \tag{2.3}$$

Then, the magnitude of the asymptotic velocity must be the same for both the inbound and outbound legs of the hyperbola:

$$v_\infty^- = v_\infty^+ = v_\infty \tag{2.4}$$

Thus, the first important result is that the modulus of the velocity relative to the planet, before and after the swing-by, remains unchanged. The swing-by can only change the direction of the relative velocity vector. Our task then is to determine, as a function of some swing-by parameter, the deflection angle δ, that is defining the rotation of $\mathbf{v}_\infty^+$ with respect to $\mathbf{v}_\infty^-$.

We can consider the polar equation of the conic section

$$r = \frac{a\left(1-e^2\right)}{1+e\cos\theta} \tag{2.5}$$

at $r \to \infty$, where $\theta = \theta_\infty$, which leads to

$$\cos(\theta_\infty) = \lim_{r\to\infty}\left[\frac{1}{e}\left(\frac{a(1-e^2)}{r} - 1\right)\right] = -\frac{1}{e}$$

or

$$\theta_\infty = \arccos\left(-\frac{1}{e}\right). \tag{2.6}$$

From the geometry in Fig. 2.6, we note that

$$\theta_\infty = \frac{\pi}{2} + \frac{\delta}{2} \tag{2.7}$$

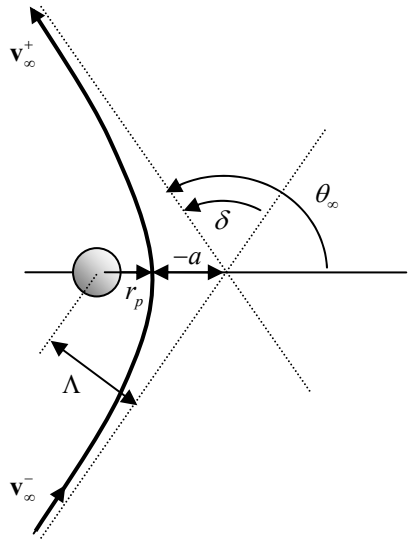


Fig. 2.6. Geometry of the unpowered swing-by.

Combining Eqs. (2.6) and (2.7) yields

$$\frac{1}{e} = \sin\frac{\delta}{2}.$$

From the energy equation (2.3), the semimajor axis of the passage is

$$a = -\frac{\mu_P}{v_\infty^2}. \tag{2.8}$$

The angular momentum can also be computed at infinity, giving

$$v_\infty \Lambda = \sqrt{\frac{\mu_P^2}{v_\infty^2}\left(e^2 - 1\right)} \tag{2.9}$$

where Λ is the distance between the planet and the asymptotes. This value can also be seen as the radial distance on the so called B-plane, i.e. the plane centred in the planet and perpendicular to $\mathbf{v}_\infty^-$. The transversal coordinate is given by the rotation of the hyperbola plane itself. Λ is very important during the operations phase, as it tells at which distance the spacecraft should target the planet, in order to achieve the desired eccentricity of the hyperbola. In the mission design phase, especially when using the patched-conic approach, Λ is not the ideal parameter to characterise the swing-by, as it is not straightforward to determine its bounds.

A better choice is the *radius of the pericentre* of the hyperbola r_p. This value is often also called *distance of closest approach*, as it represents the closest distance the spacecraft gets to the centre of mass of the planet. It is clear that if we choose r_p as a design variable, its value cannot be smaller than the radius of the planet itself, or the spacecraft will crash on the surface. If we normalise r_p using the radius of the planet R_P, we have that, for all the swing-bys,

$$r_p > 1\, R_P . \tag{2.10}$$

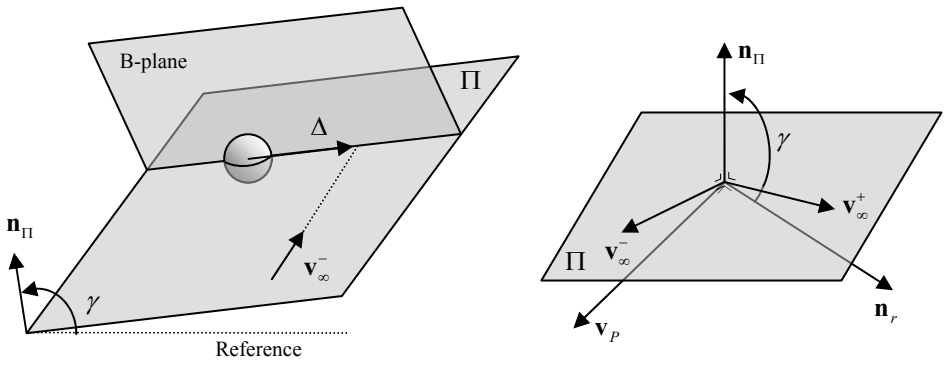
This represents a bound constraint for a design variable of the swing-by. Very often a value higher than 1 is chosen for this constraint, depending on the presence of the atmosphere, the targeting precision, and the safety margin. So minimum values ranging from 1.05 to 1.1 radii are common.

To relate the eccentricity of the hyperbola with r_p, we solve the polar equation (2.5) at periapsis, and we combine it with Eq. (2.8):

$$e = 1 + \frac{r_p v_\infty^2}{\mu_P} = 1 - \frac{r_p}{a} . \tag{2.11}$$

We have shown that during the swing-by the spacecraft follows a hyperbola, therefore the entire swing-by is planar. The hyperbola lies on the plane containing the incoming relative velocity vector and the centre of mass of the planet. This plane, though, has not been defined yet. In fact, despite the direction of $\mathbf{v}_\infty^-$ is given, and the distance Λ is a function of r_p through Eqs. (2.9) and (2.11), the transversal coordinate on the B-plane, is still undetermined. This angle defines the attitude of the swing-by hyperbola plane around the $\mathbf{v}_\infty^-$ vector (see Fig. 2.7 (a)).

This degree of freedom is due to the linked-conic approximation: basically in the heliocentric reference frame, we consider that the sphere of influence of the planet has null radius: so the spacecraft matches the planet, but the point where it pierces the sphere of influence is undetermined.



(a) Hyperbola plane (b) Geometry to compute γ

Fig. 2.7. Definition of the angle γ. (a) Shows the hyperbola plane, the B-plane and the angle γ with respect to an arbitrary reference; (b) Shows the geometry to compute γ.

To determine the plane attitude, we can provide the value of an angle γ as an additional parameter of the swing-by, with respect to a given reference direction within the B-plane, which has to be defined. Several possible choices exist for the definition of γ and the reference direction.

In this work, the attitude of the swing-by plane Π is defined by the unit vector $\mathbf{n}_\Pi$ perpendicular to the plane itself. With reference to Fig. 2.7 (b), the angle γ is the counter-clockwise angle between $\mathbf{n}_\Pi$ and $\mathbf{n}_r$, which is a unit vector perpendicular to $\mathbf{v}_\infty^-$ and the heliocentric velocity of the planet $\mathbf{v}_P$.

Given the incoming relative velocity vector $\mathbf{v}_\infty^-$ and the heliocentric velocity of the planet $\mathbf{v}_P$, the procedure for defining the vector $\mathbf{n}_\Pi$ is the following:

- Define vector $\mathbf{n}_r$, which is perpendicular to $\mathbf{v}_P$ and $\mathbf{v}_\infty^-$, i.e.:

$$\mathbf{n}_r = \frac{\mathbf{v}_\infty^- \times \mathbf{v}_P}{\left\| \mathbf{v}_\infty^- \times \mathbf{v}_P \right\|}; \tag{2.12}$$

- Rotate the vector $\mathbf{n}_r$ around vector $\mathbf{v}_\infty^-$ of the angle γ, thus finding the vector $\mathbf{n}_\Pi$.

Thus, $\mathbf{v}_\infty^+$ is found by rotating $\mathbf{v}_\infty^-$ of the deflection angle δ around the axis $\mathbf{n}_\Pi$, once δ as a function of the parameter r_p has been found through Eq. (2.11).

It shall be noted that the definition of $\mathbf{n}_r$ (and so γ) become singular if $\mathbf{v}_\infty^- \parallel \mathbf{v}_P$, due to Eq. (2.12). Therefore this model cannot be used in such a condition.

Once $\mathbf{v}_\infty^+$ has been calculated, the heliocentric velocity of the spacecraft can be computed simply by

$$\mathbf{v}^+ = \mathbf{v}_\infty^+ + \mathbf{v}_P .$$

(a) Passage in front of the planet (b) Passage behind the planet

Fig. 2.8. Two examples of heliocentric velocity change due to the swing-by in Fig. 2.6. Both cases refer to a planar case, but in (a) the spacecraft passes in front of the planet, resulting in a heliocentric deceleration, while in (b) it passes behind the planet, thus accelerating.

Although the swing-by does not change the magnitude of the relative velocity vector, there might be a significant difference in absolute velocity. This difference can be noted by considering the velocity diagrams in Fig. 2.8, referred to a planar case (i.e. all the velocity vectors lie in the same plane) for simplicity of representation. In Fig. 2.8 (a), the choice of $\gamma = \gamma^*$ is such that the spacecraft passes in front of the planet, resulting in a deceleration of its heliocentric velocity. The passage in Fig. 2.8 (b), instead, was obtained with an attitude angle $\gamma = \gamma^* + \pi$: the spacecraft passes behind the planet, and the result is an acceleration in the heliocentric reference frame. In both cases, there is a change of the heliocentric energy of the spacecraft. This change is the reason for including one or more gravity assists when planning interplanetary missions.

There are different ways of defining the attitude of the plane of the swing-by in literature; for example, Izzo [99] proposes to use the angle ζ defined through

$$\mathbf{v}_\infty^+ = v_\infty \left[\cos\delta \mathbf{b}_1 + \sin\zeta \sin\delta \mathbf{b}_2 + \cos\zeta \sin\delta \mathbf{b}_3 \right]$$

and

$$\begin{aligned} \mathbf{b}_1 &= \mathbf{v}_\infty^- / v_\infty \\ \mathbf{b}_2 &= \mathbf{b}_1 \times \mathbf{r}_P / r_P \\ \mathbf{b}_3 &= \mathbf{b}_2 \times \mathbf{b}_1 \end{aligned}$$

where $\mathbf{r}_P$ is the heliocentric position vector of the planet. Nevertheless, even this definition presents a singularity, when $\mathbf{v}_\infty^- \| \mathbf{r}_P$. It is also possible to show that, if the two definitions of the swing-by plane angle are used within an optimisation of a MGA trajectory, the complexity of the search space is the same.

We would like to compare the structure of the solution space of a simple but significant MGA transfer problem. The transfer will start with given initial conditions at Venus: after a swing-by of Venus, a deep space flight leg will reach the Earth. The trajectory is coded using the velocity formulation, and using the two T different definitions of the swing-by attitude. The parameters used in this test are

$$\mathbf{v}_0 = [-33.0145, 16.0291, 0.0110] \text{ km/s}$$
$$t_0 = 3529.8832 \text{ d, MJD2000}$$
$$T = 320.2110 \text{ d}$$

where $\mathbf{v}_0$ is the absolute heliocentric velocity at Venus, before the swing-by, t_0 is the time at Venus, T is the time of flight of the deep space flight from Venus to Earth.

The angle of the plane γ or ζ, the non-dimensional radius of the pericentre r_p, and the fraction α of at which the DSM occurs were left free, within these bounds:

$$\begin{aligned} &\gamma \in [0, 2\pi] \text{ or } \zeta \in [0, 2\pi] \\ &r_p \in [1, 3] \\ &\alpha \in [0.1, 0.9] \end{aligned} \tag{2.13}$$

Figures 2.9 from (a) to (f) show the magnitude of the DSM (i.e. cost of the transfer, or Δv) varying the geometric parameters of the swing-by, for different values of α, and for the two attitude definitions. Figures from (a) to (c) are referred to using γ, while figures from (d) to (f) to ζ.

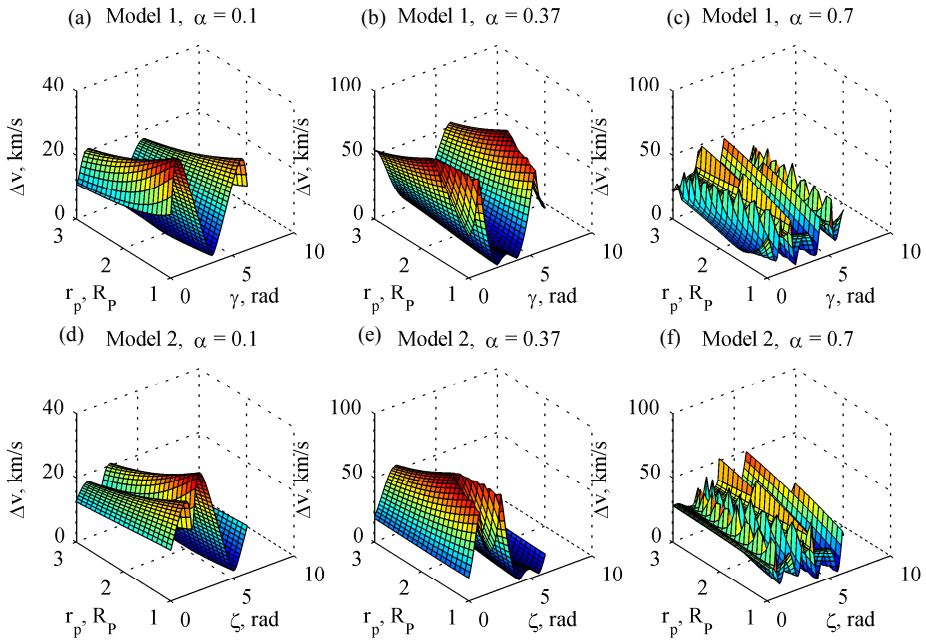
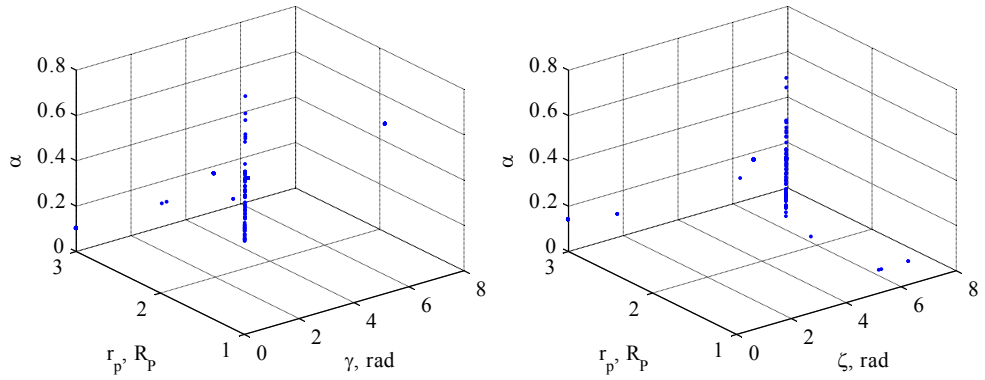


Fig. 2.9. Cost of the DSM (Δv) as a function of γ, r_p and for different values of α, by using the two definitions of swing-by plane: (a), (b), (c) using γ; (d), (e), (f) using ζ.



(a) Using γ (b) Using ζ

Fig. 2.10. Local optima of the transfer problem by using different definitions of swing-by plane: (a) using γ; (b) using ζ.

It is clear that the shape of the function in case of γ and ζ is very similar, considering that both angles are periodic over 2π, and taking into account a difference in phase of about $\pi/2$.

In order to study the distribution of the local optima in the search space (2.13), using the two different models, 200 samples has been taken. The samples were generated using a Latin hypercube distribution, and each one of them was used as a starting point for a local minimisation with the MATLAB® function *fmincon*. The local optima have been plotted in Fig. 2.10, for the two attitude models. Once again the very similar distribution of the local minima points out that there is no substantial difference in the search space, therefore the choice of one model rather than the other is irrelevant.

2.2.3 Launch

Most of the interplanetary transfer trajectories start with a launch. The launch is the manoeuvre by which the satellite is put in orbit by a launcher.

There are essentially two types of launch for an interplanetary journey. In the first one, the launcher puts the spacecraft directly into a hyperbolic escape orbit (as it happened for the Voyager spacecraft [19]); in the second type, the launcher puts the spacecraft, possibly with an upper stage, into an elliptic orbit around the Earth. Then, the firing of the upper stage or the engine of the spacecraft provides the acceleration needed to reach the hyperbolic escape orbit (as in the case of Galileo [100].

We consider the launch as the event, which injects the spacecraft into the interplanetary phase, by providing an excess of velocity, in the interplanetary frame, with respect to the departure planet. So we do not deal with the planetocentric phase of the launch, and we are not interested in whether the excess velocity is achieved through the launcher, the upper stage or the spacecraft.

This assumption is very similar to the swing-by case, in which we assumed that the velocity at infinity (asymptotic velocity) in the relative reference frame is the relative velocity that the spacecraft has at the planet in an interplanetary frame.

In addition, it is common to find the maximum launch excess velocity in the launcher manual, as a function of the spacecraft weight and declination of the escape asymptote.

In the velocity formulation of the trajectory model, the launch velocity has to be specified. Different reference frames, relative to the planet, exist to specify it. We have chosen a local spherical reference frame. With reference to Fig. 2.11, the launch relative velocity $\mathbf{v}_0$ is specified with its modulus, v_0, and with its direction, through the angles θ and δ. They refer to a local coordinate frame along the orbit of the planet: θ is the counter-clockwise angle (right ascension) in the orbital plane measuring the angle of the projection of $\mathbf{v}_0$ on the orbital plane starting from the heliocentric velocity of the planet $\mathbf{v}_P$; δ is the out-of-plane angle (declination) of $\mathbf{v}_0$ with respect to the orbital plane. Therefore, the velocity vector $\mathbf{v}_0$ in a tangential, normal, out-of-plane ($\hat{\mathbf{t}}, \hat{\mathbf{n}}, \hat{\mathbf{h}}$) reference frame (Fig. 2.11, Fig. 2.12) can be computed with

$$\mathbf{v}_{0,tnh} = v_0 \begin{bmatrix} \sin\delta\cos\theta \\ \sin\delta\sin\theta \\ \cos\delta \end{bmatrix},$$

and from this reference frame and knowing position and velocity of the planet, it is possible to transform it into inertial Cartesian reference frame, through a standard transformation. It follows that $\delta \in [-\pi/2, \pi/2]$, while for θ any period of 2π could be used, for example $\theta \in [0, 2\pi]$ or $\theta \in [-\pi, \pi]$.

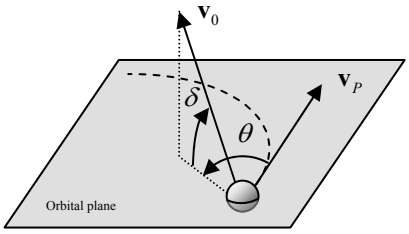


Fig. 2.11. Geometry of the launch.

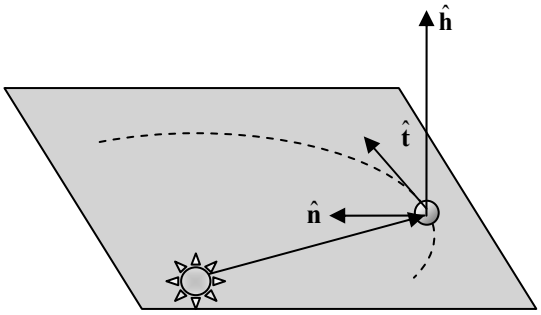


Fig. 2.12. Launch excess velocity reference frame.

The three variables for defining $\mathbf{v}_0$ are usually part of the design parameters. There are several advantages in defining $\mathbf{v}_0$ through θ and δ: first of all, the intrinsic reference frame allows not to depend on the planetary position and velocity at launch. Second, the modulus of the excess velocity, which is usually constrained by launcher capabilities, and part of the cost of the launch, is explicitly one of the parameters (therefore it is straightforward to put bound constraints on it). The direction of $\mathbf{v}_0$ can also be easily constrained, following the specifications of the asymptote direction of the considered launcher.

Assuming that the launch takes place from a given planet P_1 (usually the Earth) at time t_0, the ephemeris of the planet will provide its heliocentric position and velocity $\mathbf{x}_P, \mathbf{v}_P$. Thanks to the patched-conic approximation, the heliocentric position of the spacecraft is also $\mathbf{x}_P$, while its velocity is

$$\mathbf{v}_{0,abs} = \mathbf{v}_0 + \mathbf{v}_P .$$

UNIFORM SAMPLING

If we consider the modulus of the launch velocity v_0 fixed, then all the possible launch directions can be identified on the surface of a sphere. The drawback of considering spherical coordinates is that, when a uniform random sampling is performed on the two angles, the resulting points are not uniformly distributed on the surface of the sphere, but are concentrated on its poles (as in Fig. 2.13 (a)). This is due to the fact that the infinitesimal surface element $\mathrm{d}A$ on the sphere is

$$\mathrm{d}A = \left(v_0\,\mathrm{d}\delta\right)\left(v_0\cos\delta\,\mathrm{d}\theta\right) = v_0^2\cos\delta\,\mathrm{d}\delta\,\mathrm{d}\theta . \tag{2.14}$$

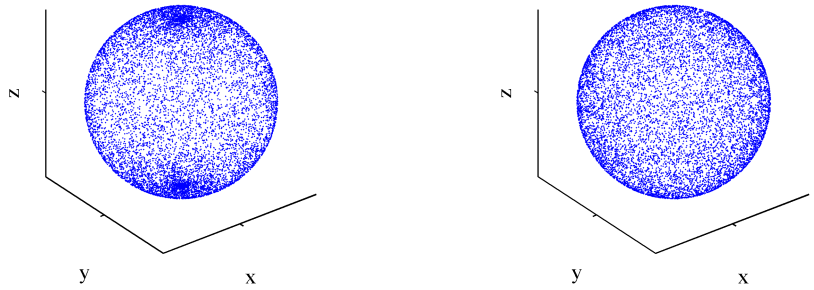
Because of to the term $\cos\delta$, for equal values of $d\delta, d\theta$, the area is smaller on the poles. This reflects in the fact that, if the coordinates δ, θ are used to sample all the possible launch directions (for example during an optimisation), then it is more likely to sample cases with polar launch. This is, actually, the least real case, as in most of the times, the excess velocity vector for an interplanetary mission lies very close to the ecliptic.

To counteract this problem, we apply a transformation to the two angles [101]:

$$\begin{aligned} \bar{\theta} &= \frac{\theta}{2\pi} \\ \bar{\delta} &= \frac{1-\sin\delta}{2} \end{aligned} \tag{2.15}$$

from which, after differentiating,

$$\begin{aligned} \mathrm{d}\theta &= 2\pi\,\mathrm{d}\bar{\theta} \\ \mathrm{d}\delta &= -\frac{2\,\mathrm{d}\bar{\delta}}{\cos\delta} \end{aligned}$$



(a) Spherical coordinates (b) Uniform distribution

Fig. 2.13. 10000 random points sampled on the surface of a sphere. (a) Sampling using spherical coordinates; (b) Sampling using uniform distribution. In (a) the points are clearly concentrated at the poles of the sphere.

And after substituting back in Eq. (2.14) we get

$$\mathrm{d}A = -4\pi v_0^2 \,\mathrm{d}\bar{\theta}\,\mathrm{d}\bar{\delta}\,,$$

from which we note that now $\mathrm{d}A$ lost the dependency on both the angles $\bar{\theta}$ and $\bar{\delta}$. By substituting the bounds of θ and δ in Eqs. (2.15) we found the bounds for $\bar{\theta}$ and $\bar{\delta}$:

$$\begin{aligned} \bar{\theta} &\in [0,1] \\ \bar{\delta} &\in [0,1] \end{aligned} \tag{2.16}$$

By uniformly random sampling the space (2.16) the surface of the sphere is also uniformly sampled, as can be seen in Fig. 2.13 (b).

2.2.4 Overall Trajectory Parameterisation

The overall trajectory in velocity formulation is made of a sequence of deep space legs alternated by swing-bys, until reaching the last planet in the sequence. Each event, either swing-by or deep space leg, is computed using the final state of the spacecraft at the previous leg, therefore the trajectory is computed following the temporal sequence of the events.

Given a sequence of n_P planets, $\mathbf{s} = \left[P_1, P_2, ..., P_{n_P}\right]$, normally the trajectory begins with a free launch at P_1 as it was shown in Section 2.2.3. The planets $P_2, ..., P_{n_P-1}$ are used for swing-bys. The trajectory ends with the last leg arriving at P_{n_P}. If planet P_{n_P} is the mission goal, then no swing-by is performed. Instead, the arrival conditions can be used to compute the cost of a brake manoeuvre or orbit injection. In any case, the trajectory has $n_P - 1$ interplanetary legs.

Other types of departure conditions are possible: a different solution vector for the trajectory is associated to each one of these. In the following subsections, three different situations are shown.

INITIAL FREE LAUNCH

This case refers to a trajectory starting with a free launch. When we say *free* we refer to the fact that the three components of the velocity vector at departure are design parameters, or somehow provided as parameters. $\mathbf{v}_0$ can be specified in any set of coordinates, but we assume to use what presented in Section 2.2.3. In any case, the launch velocity requires three variables. The departure time t_0 is another variable. Position, velocity and time are now known to initiate the interplanetary leg. This requires the time of flight to the next planet, T_1, and the timing of the DSM α_1. The following swing-by requires the plane attitude angle γ_1 and the radius of pericentre $r_{p,1}$. After the launch and the first interplanetary leg, each additional i^{th} couple of swing-by/interplanetary leg requires 4 additional parameters: $\gamma_i, r_{p,i}, T_{i+1}, \alpha_{i+1}$. The resulting solution vector, coding the design variables, is

$$\begin{aligned}\mathbf{x} = \Big[& t_0, v_0, \overline{\theta}, \overline{\delta}, \alpha_1, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, ..., \\ & \gamma_i, r_{p,i}, \alpha_{i+1}, T_{i+1}, ..., \gamma_{n_{legs}-1}, r_{p,n_{legs}-1}, \alpha_{n_{legs}}, T_{n_{legs}} \Big]\end{aligned} \tag{2.17}$$

INITIAL BALLISTIC ARC

A different way of departure can be derived from (2.17). If the first deep space flight leg has no DSM, then it can be represented with a Lambert arc only, instead of a propagated arc followed by a Lambert arc. This assumption allows a considerable reduction of the parameters in the solution vector; furthermore, the corresponding trajectory model can be obtained as a sub-case of the free launch case, without any substantial modification to the trajectory model, in the way which will be shown.

In order to solve the first interplanetary transfer entirely with a Lambert arc, we can fix $\alpha_1 = 0$ in (2.17). The propagation arc after the launch stops after $\alpha_1 T_1 = 0$, which means that there is no propagation at all. As a consequence, the launch excess velocity is irrelevant to the trajectory. In addition, if we fix $v_0 = 0$, the heliocentric velocity of the spacecraft is the same of the velocity of the planet. Therefore, the first DSM will be computed by the difference in velocity between the boundary of the Lambert arc and the planet. In other words, the first DSM becomes the launch itself, with the advantage that the variables $\overline{\theta}, \overline{\delta}$ are unnecessary, and v_0, α_1 are null.

In conclusion, for the case of a ballistic first arc, the solution vector is:

$$\mathbf{x} = \left[t_0, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, ..., \gamma_i, r_{p,i}, \alpha_{i+1}, T_{i+1}, ..., \gamma_{n_{legs}-1}, r_{p,n_{legs}-1}, \alpha_{n_{legs}}, T_{n_{legs}} \right] \tag{2.18}$$

It was found that a DSM in the leg following the launch is not essential, unless the following swing-by is resonant [102]. If this is not the case, then the

contribution of the DSM can be included in the launch, without losing good solutions, but at the same time reducing the dimensionality of the problem and simplifying the search space. If on the other hand the first swing-by is resonant, the DSM has the important task of changing the velocity vector relative to the planet, thus enhancing the effect of the swing-by. As a rule of thumb, a DSM of magnitude Δv could result in a difference in relative velocity at the planet of about $2\Delta v$.

INITIAL SWING-BY

In some cases, the trajectory under consideration is only part of the entire mission transfer, therefore the initial state could be given as swing-by incoming condition. Then the first event in the trajectory is a swing-by of P_1. Given that the initial absolute velocity $\mathbf{v}_0$ or $\mathbf{v}_{0,abs}$ is assigned, as well as the initial time t_0, the solution vector $\mathbf{x}$ is essentially the same as in the general case (2.17), removing the variables related with launch and first leg:

$$\mathbf{x} = \left[\gamma_1, r_{p,1}, \alpha_1, T_1, ..., \gamma_i, r_{p,i}, \alpha_i, T_i, ..., \gamma_{N_L}, r_{p,N_L}, \alpha_{N_L}, T_{N_L} \right] \tag{2.19}$$

2.2.5 Discussion

We present here a brief discussion on two important points of the MGA trajectory model: the first is about the complexity of this model, which generates an exponential growth of the number of solutions as the number of legs increase; the second is about the possible strategies for obtaining interplanetary legs with more than one full revolution.

MODEL COMPLEXITY

The velocity formulation of the trajectory solves explicitly the gravity assist constraints. To do that, it requires the incoming velocity before computing the outgoing velocity. Furthermore, the computation of the transfer arc from planet P_i to DSM at position M_{i+1} requires the velocity at the beginning of the arc. As a consequence, the Lambert arc from M_{i+1} to P_{i+1} is dependent on the full state vector at the end of the preceding arc.

In general, every event coded using the velocity formulation (unpowered swing-by and interplanetary leg) depends on the variables of $\mathbf{x}$ which compete to that event, but also on the initial conditions. These in turn depend on all the preceding variables in $\mathbf{x}$. In other words, velocity formulations suffer from a dependency problem: each stage of the trajectory cannot be computed without all the preceding events. If we attempt to discretise the variables in the solution vector, the number of possible trajectories, corresponding to each discrete set of variables, grows exponentially with the number of legs.

MULTIPLE REVOLUTIONS

By using the deep space flight leg as described for the velocity formulation (i.e. propagation arc followed by a Lambert arc), multiple revolutions around the main attractor can be obtained essentially in three ways.

The first is by using the propagated arc, before the DSM. In fact, if the time of propagation $\alpha_i T_i$ is long enough, the spacecraft will eventually perform one or more full revolutions on its elliptic orbit. In this way, there is no direct control of the number of revolutions, as it depends on the period of the current orbit of the spacecraft, which in turn depends on all the previous parts of the trajectory.

The second way of having multiple revolutions in a leg, is by explicitly solving a multiple revolution Lambert problem. In this case, the number of full revolutions shall be specified, and in fact this introduces a discrete variable for each Lambert arc in the design parameters. The number of full revolutions is directly controlled, and they will be performed after the DSM.

Finally, more than one full revolution (but less than two) can be obtained even if the propagated arc and the Lambert arc sweep an arc smaller than 2π individually, but they achieve a rotation greater than 2π when joined together.

Certainly these three situations can be combined together to obtain a multiple revolution leg with more full revolutions both before the DSM and after the DSM.

One could wonder, given an interplanetary leg with several full revolutions, in which revolution the DSM shall be placed. If the DSM does not change the semimajor axis of the orbit (and thus the period), then it is irrelevant whether it occurs at the first revolution or at the last one: the final state of the spacecraft at the end of the leg will be the same. This is not true if the DSM changes the semimajor axis: in this case, the spacecraft will cover more or fewer revolutions on the orbit after the DSM, which has a different period, and so the final state of the spacecraft will be different.

2.3 Position Formulation

The *position formulation* originated from the idea of keeping all the coast arcs of the trajectory uncoupled: position and time of each event along the trajectory are assigned: DSMs and swing-bys, and then each arc is computed as a solution of Lambert's problem. Consecutive deep space flight legs are matched together at the planet through a powered swing-by event. Therefore, interplanetary legs are computed first, and their velocities at the boundaries are used to compute the swing-by.

2.3.1 Interplanetary Leg

Each planet-to-planet interplanetary leg in this model is composed of a number of coast arcs, connected by DSMs.

For each DSM, the time and the position are free parameters of the model. With reference to Fig. 2.14, also the departure time from planet P_1 and the arrival time at planet P_2 are free parameters of the model, and from these free parameters, through the ephemerides, it is possible to compute the position and velocity of the planets on their orbits. The position and velocity of the planets is important to

compute the incoming and outgoing conditions of the gravity assist manoeuvre according to the powered swing-by model, as it will be explained in Section 2.3.2.

The position of each DSM, in a given leg, is identified through its radius, right ascension and declination defined in a particular reference frame, centred in the main attractor. With reference to Fig. 2.15, let assume a leg starting from planet P_1 at time t_1 to planet P_2 at time t_2. Let us consider, as reference plane, the plane defined by the orbit of planet P_1. The position of the i^{th} DSM, which occurs at point M_i in space, is specified by the following components:

- Distance from the main attractor, r_{M_i};
- In-plane angle (right ascension) θ_{M_i}, counter-clockwise in the reference plane, and measured from planet P_1 at the time of planetary encounter;
- Out-of-plane angle (declination) φ_{M_i}, or elevation angle from the reference plane.

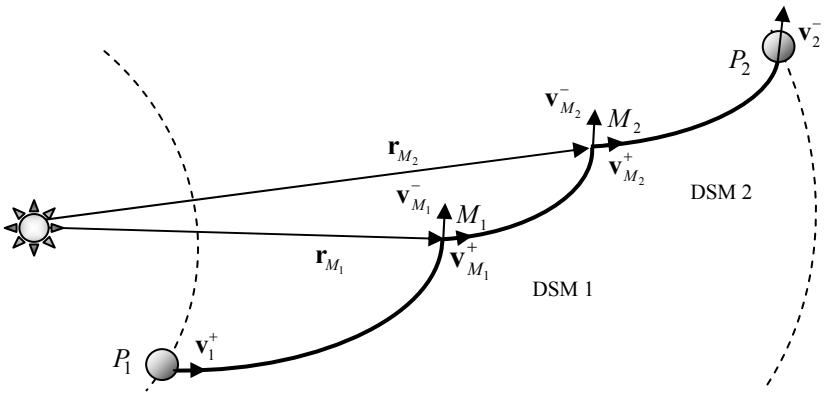


Fig. 2.14. A representation of a leg (with two DSMs) in the position formulation.

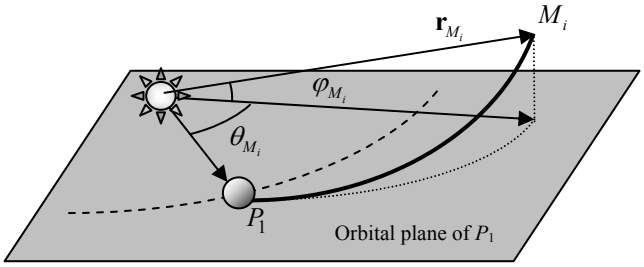


Fig. 2.15. Definition of the position of the DSM.

This choice is made such that the position of the DSM depends on the position on the previous planet. Since usually the DSM is more efficient if it follows closely the swing-by, then box bounds can be easily defined to constrain the DSM to this position relative to the planet. On the other hand, the position of the DSM changes with the time of the swing-by, following the ephemeris, and thus the three coordinates are not sufficient to determine its position in space. Assigning the position and epoch of the DSMs is equivalent to assigning the epoch of the swing-bys and then obtaining their positions through the ephemerides.

The radial distance r_{M_i} can be given non-dimensionally or dimensionally. In the non-dimensional case, the following choice was made:

- $r_{M_i} = 0$ corresponds to the radial distance of the first planet r_{P_1} at the time of the encounter;
- $r_{M_i} = 1$ corresponds to the radial distance of the second planet r_{P_2} at the time of the encounter.

Note that this does not mean that suitable values for the non-dimensional radial distance r_{M_i} are bounded in the interval $[0,1]$.

The choice of non-dimensional radius once again turns out to be useful for setting proper bounds for the variable: in fact, the justification for this type of normalisation can be found by considering the common case in which the orbits of P_1 and P_2 are circular or quasi-circular. If the transfer between the two is similar to a Hohmann type of transfer, then the trajectory will be confined in the circular corona delimited by the orbits of the two planets, and so will the DSMs. Thus in this case the bounds for the non-dimensional $\mathbf{r}_{M_i}$ can be set to $[0,1]$ or tighter.

In all the other cases, especially if the planet's orbits are eccentric, values of the radial distance less than 0 or greater 1 are perfectly acceptable.

Note that the main drawback of using a non-dimensional radius is when the transfer trajectory lays well outside the corona defined by the planetary orbits. This can be the case for highly elliptical orbits, but especially for resonant legs. In a resonant leg, the departure and arrival planets are the same, thus the corona between the two orbits degenerates, and usually the trajectory of the spacecraft goes quite far from it. In addition, the radius of the planet at departure r_{P_1} will be very similar (in the case of non-eccentric planets) to the radius of the planet r_{P_2}. The difference between the two then is unsuitable to be used for normalisation. In this case, then, it is advisable to use the dimensional distance.

The time of each DSM t_{M_i} is defined as a fraction of the total time of flight of the leg, by using the parameters $\alpha_{M_i} \in [0,1]$:

$$\alpha_{M_i} = \frac{t_{M_i} - t_1}{t_2 - t_1}.$$

Once the position and timing of all the DSMs in the leg have been defined, as well as the position and timing of the planetary encounters, then it is possible to

solve all the coast arcs. In fact, each coast arc can be computed as the solution of a Lambert's problem. The first coast arc connects P_1 at t_1 with M_1 at t_{M_1}, and its duration is $(t_2 - t_1)\alpha_1$. If n_{DSM} DSMs are present, then each arc connecting two consecutive DSMs M_i and M_{i+1} has a duration of $(t_2 - t_1)\alpha_{M_{i+1}} - t_{M_i}$. Finally, the last arc connects the last DSM to P_2 at time t_2, and lasts $t_2 - t_{M_n}$.

Note that, if multiple revolution and/or retrograde Lambert arcs have to be considered, then additional parameters are needed to specify the type of each Lambert arc.

As a result of the solution of the Lambert problems, a set of velocities are obtained. In particular, the velocity $\mathbf{v}_1^+$ at planet P_1, the velocity $\mathbf{v}_2^-$ at planet P_2, and all the pairs of velocity vectors across each DSM, $\mathbf{v}_{M_i}^-$ and $\mathbf{v}_{M_i}^+$. These can be used to compute the total cost of the interplanetary leg. In fact, each engine burn at the generic DSM point M_i must provide a change in velocity such that $\mathbf{v}_{M_i}^-$ turns into $\mathbf{v}_{M_i}^+$. Therefore, the total cost Δv of the interplanetary leg is given by:

$$\Delta v = \sum_{i=1}^{n_{DSM}} \left| \mathbf{v}_{M_i}^+ - \mathbf{v}_{M_i}^- \right|.$$

If the transfer leg is the first in a sequence, its initial velocity vector is used to compute the launch conditions, while if the leg is the last in a sequence its final velocity vector is used to compute the capture conditions. The velocity vectors at the boundaries of each other transfer leg are matched to the velocity vectors at the beginning or at the end of, respectively, the following or preceding leg through a powered swing-by.

2.3.2 Powered Swing-by

A powered swing-by manoeuvre exploits a combination of gravity and propulsive action to turn the relative incoming velocity vector onto the relative outgoing velocity vector [59]. The legs preceding and following the swing-by provide the incoming velocity $\mathbf{v}^-$ and the outgoing velocity $\mathbf{v}^+$.

Assuming that the spacecraft approaches the planet from infinity ($r \simeq \infty$), and considering that the planet moves with velocity $\mathbf{v}_P$, the relative incoming and outgoing velocities $\mathbf{v}_\infty^-$ and $\mathbf{v}_\infty^+$ can be computed as $\mathbf{v}_\infty^{+/-} = \mathbf{v}^{+/-} - \mathbf{v}_P$. Since $\mathbf{v}_\infty^-$ and $\mathbf{v}_\infty^+$ are computed independently one from the other, they do not necessarily have the same modulus and the gravity of the planet cannot be sufficient to turn the incoming asymptote onto the outgoing one. Therefore, an impulse manoeuvre is introduced at the pericentre of the inbound hyperbola leg (denoted with (1) in Fig. 2.16) to generate an outbound hyperbola leg (denoted with (2) in Fig. 2.16) with the required $\mathbf{v}_\infty^+$. Since the velocity is maximum at pericentre, a manoeuvre in this point represents the most effective way to achieve the outgoing conditions [103].

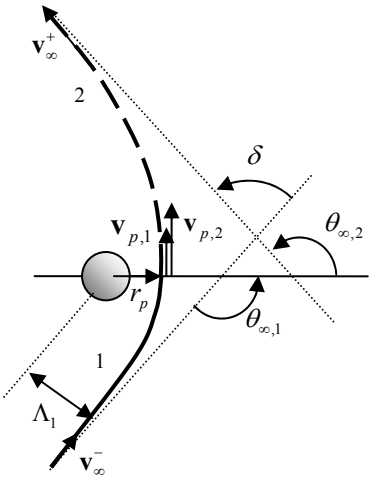


Fig. 2.16. Geometry of the powered swing-by with tangential manoeuvre at pericentre.

The required deflection angle, δ, is defined as the angle between $\mathbf{v}_\infty^-$ and $\mathbf{v}_\infty^+$:

$$\delta = \frac{\arccos\left(\mathbf{v}_\infty^+ \cdot \mathbf{v}_\infty^-\right)}{\left|\mathbf{v}_\infty^+\right|\left|\mathbf{v}_\infty^-\right|}.$$

In theory, a single manoeuvre at the pericentre of the hyperbola, at distance r_p, normal to the local position vector and in the plane of the hyperbola, should be sufficient to achieve the required outgoing velocity vector. However, the distance r_p is limited to be above the surface of the planet (or above its atmosphere). The set of possible outgoing velocities is therefore limited. If no tangential manoeuvre can achieve the desired outgoing conditions, a non-tangential manoeuvre is required. In the next two sections, the two models for tangential and non-tangential manoeuvre are presented.

TANGENTIAL MANOEUVRE

If the manoeuvre is tangential to the orbit at pericentre, it does not change the position of the pericentre (see Fig. 2.16). Using the results found for the unpowered swing-by, the semimajor axes of the two legs are:

$$a_1 = -\frac{\mu_P}{\left(v_\infty^-\right)^2}, \quad a_2 = -\frac{\mu_P}{\left(v_\infty^+\right)^2}$$

and the corresponding eccentricities, expressed as a function of the radius of the pericentre r_p are:

$$e_1 = \frac{a_1 - r_p}{a_1}, \quad e_2 = \frac{a_2 - r_p}{a_2}. \tag{2.20}$$

The problem then is to find the radius of the pericentre r_p (or equivalently, since v_∞ is given, the targeting distance Λ_1) such that the required deflection angle δ is achieved. In particular, considering Fig. 2.16, we can state that the semi-deflection of each leg of the hyperbola is related to its eccentricity in the following way:

$$\frac{\delta_1}{2} = \arcsin\left(\frac{1}{e_1}\right), \quad \frac{\delta_2}{2} = \arcsin\left(\frac{1}{e_2}\right). \tag{2.21}$$

Thus, the total deflection angle should be:

$$\frac{\delta_1}{2} + \frac{\delta_2}{2} = \delta$$

or, using Eqs. (2.20) and (2.21):

$$f\left(r_p\right) = \arcsin\left(\frac{a_1}{a_1 - r_p}\right) + \arcsin\left(\frac{a_2}{a_2 - r_p}\right) - \delta = 0\,.$$

This can be treated as a zero-finding problem, with r_p limited from below. Thus we can state the problem as:

$$\begin{aligned} &\text{Find: } r_p \mid f\left(r_p\right) = 0 \\ &\text{s.t.:} \quad r_p > r_{p,min} \end{aligned}$$

We can use a Newton-Raphson method, using $r_{p,min}$ as a starting point:

$$r_p^{(i+1)} = r_p^{(i)} - \frac{f\left(r_p^{(i)}\right)}{\left.\frac{df}{dr_p}\right|_{r_p^{(i)}}}\,.$$

If a value of r_p is found, then there is no need to investigate further. The cost of the powered swing-by in terms of Δv is:

$$\Delta v = \left|v_{p,1} - v_{p,2}\right| = \left|\sqrt{v_\infty^- + \frac{2\mu_P}{r_p}} - \sqrt{v_\infty^+ + \frac{2\mu_P}{r_p}}\right|.$$

At each iteration, we also have to monitor whether $r_p^{(i)}$ becomes smaller than $r_{p,min}$. If so, then this strategy cannot find a feasible solution to the problem (i.e., a too small value of r_p is necessary), and the loop can be aborted. Should this happen, the following strategy is started.

NON-TANGENTIAL MANOEUVRE

If a solution cannot be found with a tangential manoeuvre at pericentre, then the following strategy is used to find a suitable non-tangential manoeuvre at the pericentre of leg (1).

The value for the radius of the pericentre of leg (1) is set to $r_{p,1} = r_{p,min}$. In fact, this allows exploiting the maximum possible deviation from the natural dynamics of the swing-by, thus minimising the propelled Δv.

Since the manoeuvre is not tangential, the pericentre of leg (2) changes. With reference to Fig. 2.17, we can consider that the line of apsides of leg (2) is rotated by an angle ω with respect to leg (1). The problem in this case is to find the value of ω such that leg (2) passes through the pericentre of leg (1). The polar equation for the hyperbola leg (2) can be written as:

$$r(\theta) = \frac{a_2\left(e_2^2 - 1\right)}{1 + e_2 \cos\theta}. \tag{2.22}$$

To force the passage, we can impose that:

$$r(\omega) = r_{p,1}$$

and Eq. (2.22) becomes:

$$g(\omega) = a_2\left(e_2^2 - 1\right) - r_{p,1} - e_2 r_{p,1}\cos\omega = 0. \tag{2.23}$$

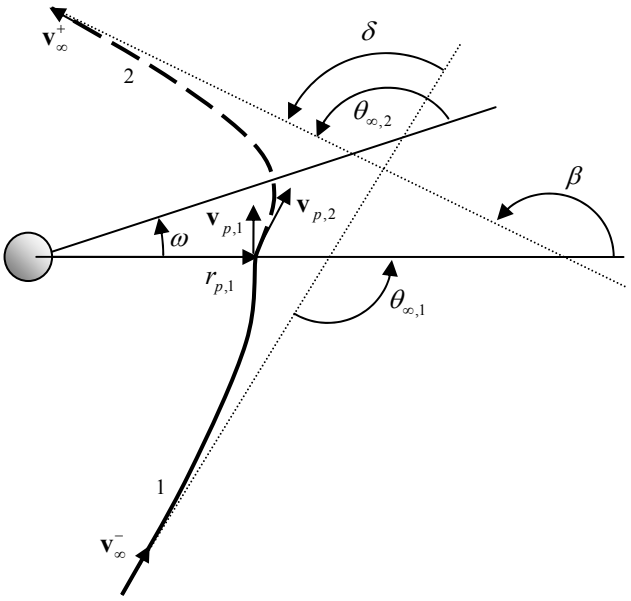


Fig. 2.17. Geometry of the powered swing-by with a non-tangential manoeuvre at pericentre.

So the problem is:

$$\text{Find: } \omega \mid g(\omega) = 0 .$$

Considering that $\theta_{\infty,2} = \beta - \omega$, Eq. (2.23) can also be rewritten as:

$$\frac{a_2}{\cos^2\left(\theta_{\infty,2}\right)} + \frac{r_{p,1}\cos\omega}{\cos\left(\theta_{\infty,2}\right)} - a_2 - r_{p,1} = 0 .$$

This time, there are no physical constraints on the value of ω, nevertheless care must be taken as there are two singularities in Eq. (2.23) for each period of ω. Once ω has been found, for example using once more a Newton-Raphson iterative method, then the Δv can be computed. The modulus of the two velocity vectors at $r_{p,1}$, before and after the manoeuvre, can be easily computed, from the energy equation, as:

$$v_{p,1} = \sqrt{\frac{2\mu_P}{r_{p,1}} - \frac{\mu_P}{a_1}}, \quad v_{p,2} = \sqrt{\frac{2\mu_P}{r_{p,1}} - \frac{\mu_P}{a_2}} .$$

The former is purely transversal, as it is at the pericentre of leg (1), while the latter is not (see Fig. 2.17). Its transversal and radial components can be calculated from the angular momentum:

$$v_{p,2,\theta} = \frac{\sqrt{\mu_P a_2\left(e_2^2 - 1\right)}}{r_{p,1}}, \quad v_{p,2,r} = \sqrt{v_{p,2}^2 - v_{p,2,\theta}^2} .$$

At this point, we can compute the total change in velocity needed:

$$\Delta v = \sqrt{\left(v_{p,2,\theta} - v_{p,1}\right)^2 + v_{p,2,r}^2} .$$

Before concluding this section, let us remark that unpowered swing-bys can be used in the position formulation. It was shown above that the powered swing-by is a manoeuvre to match the incoming and outgoing velocity vectors at the planet. If the unpowered swing-by is used, instead, Eqs. (2.4), (2.7) and (2.10) define non-linear constraints that cannot be solved explicitly, but they have to be taken into account as non-linear constraints in the trajectory model. This ensures that the modulus of the incoming and outgoing relative velocity vectors are the same, and the deflection angle δ is small enough that it can be achieved with a radius of pericentre above the lower limit, as required by the unpowered swing-by model.

2.3.3 Overall Trajectory Parameterisation

Given a sequence of planetary swing-bys, a launch planet and an arrival planet, the entire trajectory is composed of a set of legs, which connect the planets, from departure to arrival, through all the swing-bys. Each leg may contain one or more DSM.

Table 2.1. Solution vector for a trajectory in position formulation.

Parameter	Description
t_0	Launch date
T_l	Time of flight of each leg l
$\alpha_{M_i}, r_{M_i}, \theta_{M_i}, \varphi_{M_i}$	Timing and position of each DSM i

The parameters needed for modelling a trajectory with the position formulation are summarised in Table 2.1. The launch date t_0 determines when to depart from the departure planet. The timing of all the swing-bys and arrival to the last planet are defined through the time of flight of each leg T_l. The ephemerides provide the position of all the planets at the given times. Position and timing of DSMs are given through α_{M_i}, $r_{M_i}, \theta_{M_i}, \varphi_{M_i}$.

Let a trajectory be made of n_{legs} legs (and thus $n_{legs}-1$ swing-bys), with each leg l made of $n_{arcs}(l)$ arcs (and thus $n_{arcs}(l)-1$ DSMs). The total number of DSMs is

$$n_{DSM} = \sum_{l=1}^{n_{legs}} \left(n_{arcs}(l) - 1 \right),$$

and the total number of variables in the solution vector to fully characterise the trajectory is:

$$1 + n_{legs} + 4n_{DSM}.$$

The solution vector for this trajectory is therefore:

$$\mathbf{x} = \left\{ t_0, T_1, T_2, ..., T_{n_{legs}}, \right.$$
$$\left. \alpha_{M_1}, r_{M_1}, \theta_{M_1}, \varphi_{M_1}, \alpha_{M_2}, r_{M_2}, \theta_{M_2}, \varphi_{M_2}, ..., \alpha_{M_{n_{DSM}}}, r_{M_{n_{DSM}}}, \theta_{M_{n_{DSM}}}, \varphi_{M_{n_{DSM}}} \right\}.$$

Algorithm 2.1 presents briefly the loops needed to compute the entire trajectory. The overall trajectory is built by computing first all the deep space legs (and the DSMs between each couple of arcs), and then the swing-bys (and their cost).

The total cost of the trajectory is obviously the sum of all the DSMs and the swing-by impulses. The launch excess velocity and relative velocity at the target planet of the sequence can be computed taking the difference between the velocities at the bounds of the trajectory and planetary velocities.

Algorithm 2.1. Computing the entire trajectory in position formulation.

1: **For** $l = 1 \ldots n_{legs}$
2: **For** $i = 1 \ldots n_{arcs}(l)$
3: Compute Lambert arc
4: **End For**
5: **For** $i = 1 \ldots n_{arcs}(l) - 1$
6: Compute $\Delta v_{l,i}$ of the DSM between arc i and $i+1$
7: **End For**
8: **End For**
9: **For** $l = 1 \ldots n_{legs} - 1$
10: Compute powered swing-by between leg l and $l+1$
11: Find cost of the l^{th} swing-by Δv_l^{SB}
12: **End For**

2.3.4 Discussion

The position formulation model is flexible concerning the number of planetary swing-bys, as well as the number of DSMs in each leg. In fact, it is possible to vary the number of DSMs or swing-bys without any substantial change to the structure of the trajectory, and only at the cost of adding 4 more variables to the solution vector for each additional DSM, and 1 variable for each additional leg. As it will be shown in the following, especially with regard to the number of DSMs, this is something that cannot be achieved with the velocity formulation.

An advantage of this formulation is that the complexity of the MGA problem grows polynomially with the number of swing-bys and with the number of DSMs.

In the following subsections, we will briefly discuss these two features of the position formulation model. In addition, a few considerations about using powered and unpowered swing-bys are presented.

USING MORE DSMS PER LEG

There are essentially two advantages in using more than one manoeuvre in each leg. The first is that each DSM splits the Lambert arcs, thus enabling multiple revolution legs without solving a multiple revolution Lambert problem. For example, let us assume to have one leg with no DSM. If we solve the arc using a single revolution Lambert arc, then spacecraft cannot perform more than one revolution during that leg. Now imagine introducing a DSM in the leg: in this case, the leg will be composed of two single-revolution Lambert arcs, and so the spacecraft will be able to perform up to 2 complete revolutions. Note that shorter solutions are still available even with an arbitrary number of DSMs.

This can be seen as a way to tackle multiple revolution legs without the need for an integer to represent the number of revolutions in the Lambert problem, and thus avoiding to have a mixed integer-real search space.

The second advantage is that it allows a better distribution of the Δv along the leg. For some types of legs, two impulses are necessary to reach the desired target

orbit with the correct phase. In addition, if we fraction the Δv provided by a single DSM into more manoeuvres, the amount of Δv of each manoeuvre is smaller. This means that even a smaller level of thrust, hence a smaller engine, is needed to obtain the velocity change. The more DSM we consider for each leg, the more the leg approximates a low-thrust arc, in which the Δv is provided continuously and is distributed along the entire arc.

The following example illustrates how the total cost of the trajectory can take benefit from multiple DSMs. Two types of EVM transfers are considered: both of them have no DSM in the first leg; in the second leg, instead, the former has one DSM, while the latter has got two. If we identify each DSM with a *d* in the sequence, the two sequences can be identified with EVdM and EVddM, respectively.

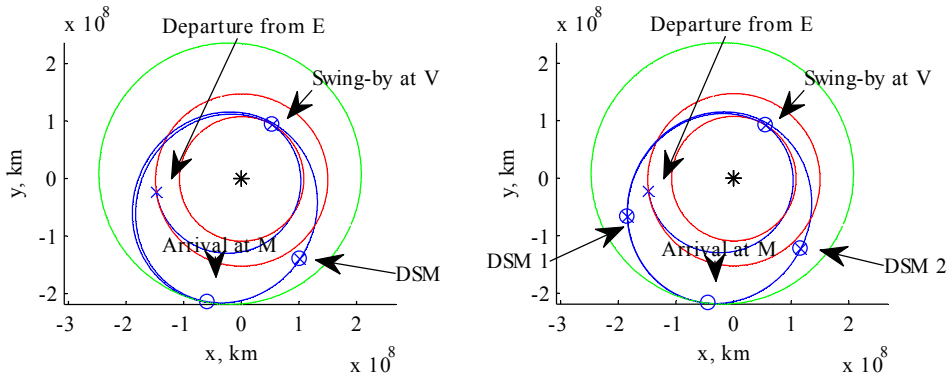
For both instances, the objective is to minimise the total Δv, which is the sum of the launch excess velocity, the relative velocity at arrival at Mars (assuming that we would like to rendezvous with Mars), and the DSMs. An optimisation can be run for the two cases, using the bounds presented in Table 2.2: the best solutions found for either case are represented in Fig. 2.18. Although the two transfers appear identical, the one with two manoeuvres is 50 m/s cheaper.

Model Complexity

As we mentioned already, one important advantage of this formulation is that the model complexity is polynomial with respect to the number of DSMs and the number of swing-bys. This is a direct consequence of the fact that the position formulation allows us to compute each arc independently of the other arcs.

Table 2.2. Bounds for EVM transfer, both in case of one and two DSMs in the second leg.

Variable	EVdM		EVddM	
	LB	UB	LB	UB
t_0, d, MJD2000	4452.5	4492.5	=	=
T_1, d	152.29	192.29	=	=
T_2, d	677.61	717.61	=	=
r_1	0	1	=	=
θ_1, rad	0	2π	=	=
φ_1, rad	–0.1	0.1	=	=
α_1	0.01	0.6	0.01	0.3
r_1	/	/	0	1
θ_1, rad	/	/	0	2π
φ_1, rad	/	/	–0.1	0.1
α_1	/	/	0.3001	0.6



(a) EvdM (b) EVddM

Fig. 2.18. Best solutions found for the two instances of the EVM transfer problem. (a) EVdM, the total Δv is 8.14 km/s; (b) EVddM, the total Δv is 8.09 km/s.

To explain better this concept, let us assume, without loss of generality, that the problem is planar and that the distance of each DSM from the centre of the coordinate system is constant; then, the position M_i of each DSM can be identified by a single variable: the angle θ_{M_i}. We can for example set $r_{M_i} = 1$ and $\varphi_{M_i} = 0$. The time at which the DSM happens requires another variable, α_{M_i}, or equivalently the time t_{M_i}. If we consider that each angle θ_{M_i} can assume only values from a set of k on the whole circle $[0, 2\pi]$, and each time a set of h values, then the number of distinct possible positions on an ideal time-space grid for a DSM is equal to hk. The position of the planets is determined through the ephemerides, thus only one parameter, the epoch, has to be specified. Once again, it is assumed that there is a finite set of h possible epochs. This means that there are

$$h \cdot hk = h^2 k$$

possible distinct arcs to go from planet P_1 (at a given time t_1) to M_1 (at a given time t_{M_i} and position θ_{M_i}). These arcs can be computed independently of the rest of the trajectory, once $t_1, t_{M_1}, \theta_{M_1}$ are given. The same holds for connecting the last DSM to the arrival planet of the leg.

An arc connecting two consecutive DSMs is determined when time and position of the two DSMs is fixed (see Fig. 2.19 for an ideal representation). Thus, the total number of independent arcs is

$$hk \cdot hk = h^2 k^2 .$$

Once again, these arcs can be computed independently of the other parts of the trajectory. For the trajectory given as an example, with 2 DSMs, the total number of independent legs is

$$h^2 k + h^2 k^2 + h^2 k = 2h^2 k + h^2 k^2$$

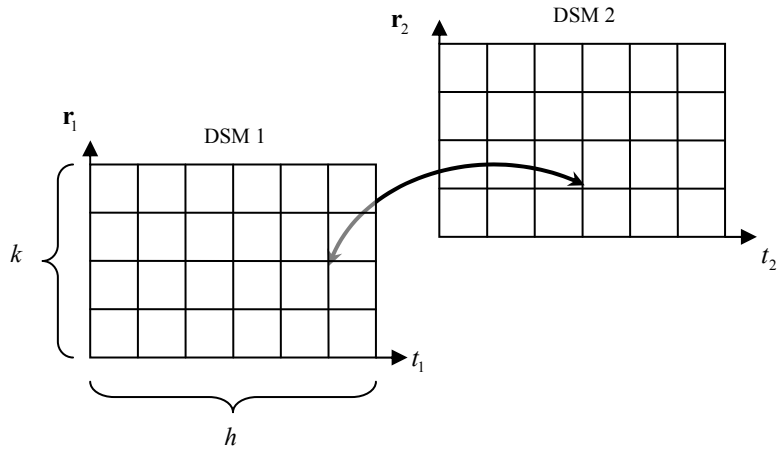


Fig. 2.19. If we imagine to discretise position and time of two consecutive DSMs, then the possible arcs connecting the two are all and only those found by connecting a point on the grid DSM 1 to a point on the grid DSM 2. The arcs do not depend on any other parameter in the solution vector.

and in general, if n_{DSM} DSMs are considered:

$$2h^2k + \left(n_{DSM} - 1\right)h^2k^2 .$$

Therefore, the position formulation does not suffer from any dependency on the previous legs and the growth of the number of solutions is polynomial.

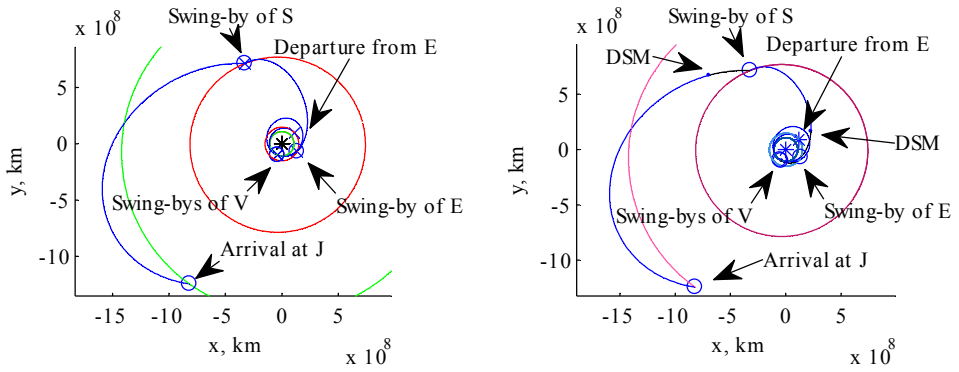
COMPARISON BETWEEN POWERED AND UNPOWERED SWING-BYS

It was mentioned in Chapter 1 that the powered swing-by generates super-optimal solutions, with respect to the model with manoeuvres in deep space only. This is due to the fact that performing a corrective manoeuvre at the pericentre of the swing-by hyperbola is more efficient than performing the manoeuvre in deep space, even if the manoeuvre is very close to the planet ($\alpha = 0$). As a result, the same transfer trajectory may result to be less expensive (in terms of Δv) when computed using powered swing-bys, than the using velocity formulation. This can be a problem because, as discussed, powered swing-bys pose serious constraints in the operations phase.

As an example, let us consider the Cassini mission. If the trajectory is modelled with the position formulation and no DSMs, the resulting minimum-Δv trajectory is shown in Fig. 2.20 (a): the total Δv, including launch excess velocity and all the powered swing-bys, resulted to be 4.45 km/s (the statement of this transfer optimisation problem, including bounds and objective function, can be found in [65]).

The same solution can be constructed using the velocity formulation with DSMs. A first guess is generated by extracting all the parameters necessary for the velocity formulation from the optimal trajectory in position formulation, except for the positions of the DSMs. These are initially set very close to the departure planet ($\alpha_i \cong 0.01$). An optimisation is then run to find a locally optimal solution in a

neighbourhood of the first guess. The resulting solution, in Fig. 2.20 (b), has a cost of 4.58 km/s, which is about 130 m/s more expensive than the analogous solution in position formulation.



(a) Position formulation (b) Velocity formulation

Fig. 2.20. ACT "Cassini1" optimal solution with position formulation (a) and velocity formulation (b).

2.4 Discussion

In this chapter, two different formulations for modelling an interplanetary MGA trajectory have been presented: the position formulation and the velocity formulation. Appendix B describes a paradigm to model virtually any type of trajectory, by decomposing it into building blocks. It is shown that the paradigm can be used to model the position and the velocity formulations of the trajectory, as well as a combination of the two. Furthermore, it is shown that it is possible to sort the blocks in a sequence such that it can be evaluated incrementally.

The incremental pruning method that will be presented in this thesis is based on the velocity formulation of MGA trajectories; some aspects of the position formulation were addressed by Becerra et al. in [104] and [105]. The main motivations for addressing the velocity formulation can be summarized as follows: the velocity formulation gives an unconstrained problem with lower dimensionality with respect to the position formulation, which in turns gives a constrained problem with higher dimensionality; the model based on the velocity formulation is closer to the actual way trajectories are operated in space (e.g. no powered swing-bys are performed in real missions); finally, as mentioned above, a model based on the position formulation can generate solutions that are more energy efficient than a model based on the velocity formulation, therefore the latter is more conservative. Nevertheless, an incremental approach based on the position formulation is possible, thanks to the findings in Appendix B.

3

INCREMENTAL PRUNING

This chapter will describe the incremental pruning process that is proposed in this thesis, to prune the search space of gravity assist trajectories. The objective of this algorithm is to remove those parts of the search space in which a global optimum is less likely to be found and to intensify the search on the remaining parts. A number of tests, based on a rigorous testing procedure, will compare the performances of the pruning approach against other deterministic and stochastic off-the-shelf optimisers.

3.1 Introduction

A single-objective global minimisation problem with bound constraints can be stated in the following way. Consider a scalar function $f(\mathbf{x})$, in which the decision (or solution) vector $\mathbf{x}$ is bounded within the set $D = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u \right\}$. The set D will be called the *solution domain* or *solution space*, in the following, and $\mathbf{x}_l$, $\mathbf{x}_u$ are the lower and the upper bounds on the values of $\mathbf{x}$.

Hence, the problem is to find $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \le f(\mathbf{x}), \forall \mathbf{x} \in D$. Equivalently, it is possible to define a global maximisation problem by defining $f' = -f$. We will refer to the minimisation problem in a compact form as:

$$\min_{\mathbf{x} \in D} f(\mathbf{x}) \tag{3.1}$$

Problem (3.1) aims at identifying a single point in *D*. A more general problem is the one of finding the ε-set containing the global optimum $\mathbf{x}^*$. The ε-set is defined as follows, given $\varepsilon > 0$:

$$X = \left\{ \mathbf{x} \in D \mid f(\mathbf{x}) \le f(\mathbf{x}^*) + \varepsilon \right\} \tag{3.2}$$

The ε-set is generally disconnected and requires the identification of multiple simply connected subsets of D. However, this second problem is particularly common in modern space mission preliminary design. In fact, considering that each variable in the decision vector represents a design parameter, having a number of different solutions $\mathbf{x}_i'$ implies having different optimal design points, among which a trade off can be performed. Considerations, related with other subsystems of the spacecraft, can be used to choose the baseline solution: for example, the trajectories may differ for their eclipses, or arrival and departure conditions, or reliability due to passages in the vicinity of bodies or radiation belts, etc. In space mission design, having one or more back-up designs is also fundamental, due to the strict conditions on launch. Other solutions can be used for designing back-up missions.

Furthermore, the ε-set contains the neighbourhoods of all the local minima at distance lower than ε from the global one. The identification of the neighbourhood provides a measure of the robustness of the local minima, i.e. a large neighbourhood corresponds to a robust local minimum. In the specific case of trajectory design, the neighbourhood is generally called a *launch window*.

Many optimisation algorithms (optimisers) exist in literature to tackle problem (3.1). Many of them consider the objective function $f(\mathbf{x})$ as a *black box*, i.e. they do not need any particular information about the shape or the properties of the function f. If the function is not defined in a subset of the domain D, a simple workaround is still possible by assigning a very high fictitious objective value. Others, like sequential quadratic programming (SQP) [106] require some assumptions on the properties of f, most commonly existence, boundedness, continuity or differentiability over the domain.

The approach proposed in this section is conceptually different from black-box approaches but still avoids the requirement on the continuity and differentiability of the objective function. The idea is to exploit some general characteristics of the problem to direct the search for the ε-set only towards promising areas of the search space, avoiding the typical exploration overhead of generic black-box algorithms. At the same time the algorithmic complexity is maintained polynomial, avoiding the exponential increase of the computational cost with the problem dimensionality (typical of exhaustive methods).

According to the well known “no free lunch” theorem [107], on a particular problem, different search algorithms may obtain different results, but over all problems, they are indistinguishable. It follows that, if an algorithm achieves superior results on some problems, it must pay with inferiority on other problems. In this sense, there is no free lunch in search.

Therefore, the idea here is to build a search method that is specialised on a specific class of problems, of which the MGA trajectory design problem is one instance, to maximise performance rather than versatility (Fig. 3.1).

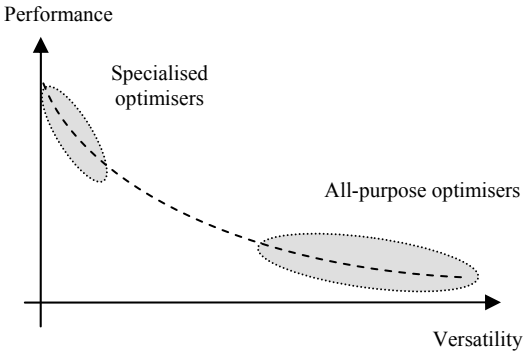


Fig. 3.1. Versatility VS. performance for optimisation methods.

3.2 Incremental Approach

3.2.1 General Concept

The main idea behind incremental approaches is to decompose a problem into a cascade of sub-problems, of smaller dimensions, and build a solution by composing the solutions of all the sub-problems.

In particular, the problem is to search for the ε-set by incrementally removing the regions of the domain in which there is no possibility to find good solutions (or there is very little). The process of removing some parts of the solution space is commonly know in literature as *pruning*, due to the fact that it is mostly used by branch and bound techniques, which explore the whole domain as a tree [78]. The final aim of the incremental process is to obtain a residual domain $\bar{D}$ after pruning. A search can be performed on the residual domain, to find the global optimum or a set of low-laying solutions. If the pruning is effective, the remaining solution space should be smaller and easier to explore, by means of any generic search engine. Thus, a lower computational effort should be needed to find a solution of the same quality with respect to the exploration of the whole domain *D*. Equivalently, better solutions can be found with the same computational effort.

The incremental pruning approach, introduced in this section, exploits important properties of the MGA trajectory design problem. However, the method itself can be applied in principle to any problem whose objective function has the same properties.

Let us consider a partitioning of the variables in the solution vector $\mathbf{x}$ into N_L partitions:

$$\mathbf{x} = \left[\mathbf{x}_{L,1} \quad \mathbf{x}_{L,2} \quad \cdots \quad \mathbf{x}_{L,N_L} \right].$$

The number of variables within each partition *i* is free, and depends on the problem through the objective function, as it will be shown later for the MGA trajectory.

Each set of variables $\mathbf{x}_{L,i}$ is bounded by $\mathbf{x}_l$ and $\mathbf{x}_u$, so all the possible points $\mathbf{x}_{L,i}$ define a hyper-rectangular domain, which will be named $D_{L,i}$, hence we can write $\mathbf{x}_{L,i} \in D_{L,i}$. Since the dimensionality of $D_{L,i}$ is smaller than the dimensionality of *D*, but the bounds for the dimensions in common are the same, we can consider $D_{L,i}$ as a *dimensional slice* of *D*, along the dimensions represented by $\mathbf{x}_{L,i}$.

We introduce now another set of vectors of variables which are built up in the following way:

$$\mathbf{x}_1 \equiv \mathbf{x}_{L,1}$$
$$\mathbf{x}_2 \equiv \left[\mathbf{x}_{L,1} \quad \mathbf{x}_{L,2}\right]$$
$$\ldots$$
$$\mathbf{x}_{N_L} \equiv \left[\mathbf{x}_{L,1} \quad \mathbf{x}_{L,2} \quad \ldots \quad \mathbf{x}_{L,N_L}\right] \equiv \mathbf{x}$$

Correspondingly, we define the set of domains D_i, such that $\mathbf{x}_i \in D_i$. Again, the domains D_i are dimensional slices of *D*. In particular, we can define each one as:

$$D_i = \prod_{k=1}^{i} D_{L,k}, \quad i = 1 \ldots N_L,$$

where the product is to be intended as a Cartesian product among sets. Note that, in the particular case of $i = N_L$, we have $D_{N_L} \equiv D$.

The problem has now been decomposed into sub-problems, which will be referred to as *levels* in the following. Note that the sub-problem at each level includes all the variables of the preceding levels.

Now let us introduce an objective function $f(\mathbf{x})$, and assume that the function is a sum of terms f_i which depend on $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_L}$:

$$f(\mathbf{x}) = \sum_{i=1}^{N_L} f_i(\mathbf{x}_i). \tag{3.3}$$

We will call each function f_i *partial objective function* for level *i*. Thus, the total objective is built-up level by level, incrementally. It is important to stress that the function f_i associated with level *i* depends only on the part of the solution vector related to the levels from 1 to *i*.

The underlying assumption of the incremental approach is that the solution of the whole problem is the sum of the solutions of all the sub-problems. In order to satisfy this assumption the functions f_i need to be properly defined. Without loss of generality, let us consider that each f_i is non-negative:

$$f_i(\mathbf{x}_i) \geq 0, \ \forall \mathbf{x}_i \in D_i, i = 1 \ldots N_L. \tag{3.4}$$

According to Bellman's principle of optimality [108], if all the partial solutions from 1 to *i* are optimal, f_i is a lower bound for f_j, when $j > i$, and for the whole objective function f. Although this is generally true, it does not help us to define a proper partial objective function since a minimum, local or global, of the partial objective function is not generally a minimum for the whole objective function f. On the other hand, if $\mathbf{x}^*$ is the global minimum of the objective function *f*, we can identify at each level *i* a set $\bar{D}_i$ such that $\mathbf{x}_i^* \in \bar{D}_i$, where $\mathbf{x}_i^*$ is the partial vector containing the components of $\mathbf{x}^*$ up to level *i*. We start by defining a subset of $D_{L,i}$ as:

$$\bar{D}_{L,i} = \left\{ \mathbf{x}_{L,i} \in D_{L,i} \mid f_i\left(\left[\mathbf{x}_{i-1} \quad \mathbf{x}_{L,i}\right]\right) \le \bar{f}_i \right\} \tag{3.5}$$

where $\bar{f}_i$ is the *pruning threshold*. Then we can define the *feasible set* $\bar{D}_i \subseteq D_i$ at level *i* as:

$$\bar{D}_i = \bar{D}_{i-1} \times \bar{D}_{L,i}\,. \tag{3.6}$$

We call the Boolean condition

$$\Phi_i(\mathbf{x}_i) = f_i \le \bar{f}_i \tag{3.7}$$

the *pruning criterion*, since it generates the domain $\bar{D}_i$ by removing, or pruning, the portion of D_i that does not satisfy the condition. The interest is therefore to converge to a set of solutions with a low laying value of the partial objective function. Note that the definition of the feasible set (3.6) is consistent with Bellman's principle of optimality. Once $\bar{D}_i$ has been identified, through a search method, we can consider for level $i+1$ the new solution space:

$$\bar{D}_i \times D_{L,i+1}\,. \tag{3.8}$$

The overall process is called *incremental pruning* and requires the definition of a pruning criterion at each level *i*. What makes this approach interesting is that the evaluation of a partial objective function can be remarkably less expensive than the evaluation of the complete function *f*, and the associated search space is easier to explore. Thus it is possible to search on level 1, using f_1 on $D_1 \equiv D_{L,1}$, and ideally remove (or prune) from the search space all the sets of values for which the partial objective function is above the threshold. The result is a residual partial domain $\bar{D}_1 \subseteq D_1$. Then the process continues with level 2, considering f_2, on $\bar{D}_1 \times D_{L,2}$. Note that this partial domain has a smaller volume than $D_1 \times D_{L,2}$, as there are sets of points in D_1 which have already been discarded during the pruning of level 1. The reduction of the search space at level *i* makes the search at level $i+1$ more effective. At the last level, the complete objective function f is then minimized, on

the remaining part of the search domain which was not pruned at previous levels, which is $\bar{D}_{N_L} \equiv \bar{D}$.

Note that care must be taken in defining the different pruning criteria and the partial objective functions. For example, if a threshold is used, while keeping the thresholds too high will result in a light pruning with little improvements on the computational speed of the optimiser, lowering the threshold too much may result in having the optimal solution left out of the residual search space, if the search is not exhaustive. Furthermore, as it will be shown later in different test cases, for some particular kinds of problems, the partial objective functions f_i and the pruning thresholds are not related to *f*. In fact, for some cases, it is possible to exploit the knowledge of the physics of the problem, to create a partial objective function in which f_i does not contribute to the value of the objective function of the whole problem (as in Eq. (3.3)) but is specifically devised to prune the search space at level *i*.

Note that at this point there is no assumption on the morphology of the residual domains $\bar{D}_i$: they are simply a subset of the corresponding domains D_i. Also, $\bar{D}_i$ is not necessarily simply connected.

Another important point to underline is that the proposed incremental approach, in its basic form, is independent from the search method that is used to define the feasible set $\bar{D}_i$ at each level *i*.

Finally, it is worth underlining that the incremental pruning does not change the nature of the problem. If the problem is NP-hard with the number of variables, it will remain so even with an incremental pruning. On the other hand, the algorithmic complexity remains polynomial as long as the identification of $\bar{D}_i$ is polynomial, i.e. avoids the exponential growth of the size of the solution space, as more levels are added to the problem. This can be achieved with a combination of suitable pruning criteria and polynomially complex search algorithms at each level. Part of this work will be dedicated to the study of appropriate pruning procedures, in order to maximise the pruned space, while preserving the promising solutions.

3.2.2 Back Pruning

From Eqs. (3.5) and (3.6), it is clear that the pruning process, at each level, acts on the dimensional slice of domain that includes the variables of that level. In other words, the pruning at level *i* generates a residual domain $\bar{D}_{L,i} \subseteq D_{L,i}$, but does not modify the part of the search space that involves the previous levels, i.e. $\bar{D}_{i-1}$.

This can result in a important limitation. In fact, it can happen that, while applying the pruning criterion at level *i*, a portion of the solution space, which was feasible at previous levels, does not contain any acceptable solution anymore. Hence, it is possible to further restrict the feasible solution space at level *i*. This can be achieved by defining the feasible set at level *i* directly from the pruning criterion. So instead of using Eqs. (3.5) and (3.6), we define directly:

$$\bar{D}_i = \left\{ \mathbf{x}_i \in D_i \mid \Phi_i\left(\mathbf{x}_i\right) \right\} \tag{3.9}$$

and since $\mathbf{x}_i \in D_i$, it results that $\bar{D}_i \subseteq D_i$, hence the search space can be further pruned at later levels on all the variables.

This procedure will be referred to as *back pruning* in the following, because of the need to go back to all the previous levels and redefine the feasible set. This is opposed to the incremental (forward) pruning, which prunes the search space going forward level by level, and every time freezing the domain $\bar{D}_i$.

3.2.3 Generalisation of the Incremental Pruning

The concept of the incremental pruning presented in Section 3.2.1 is exact, in the sense that, if the search for the feasible sets at each level is exhaustive, the algorithm guarantees to preserve all the solutions below the selected pruning threshold, and to prune all the others at some level of the incremental pruning process.

While applying the process to real problems, it resulted that more flexibility was needed, together with the relaxation of some of the hypotheses on the objective function or the pruning criteria.

In fact, for a wide range of real problems, the pruning criterion is not simply an upper bound on the partial objective function, as in Eq. (3.7). Very often there is the need to prune the domain on the basis of a criterion that is different from the objective function.

More generally, it is possible to prune the solution space more effectively if the pruning criterion is different from the objective function used to identify the sub-domains $\bar{D}_i$. The reason for using a different function for searching and pruning will be more clear in the following, and it is due to the fact that solution subsets can be removed not only because of the high value of f_i, but also because of other additional undesired properties.

The most noticeable example of this situation is when, still willing to minimise the Δv, some solutions have a transfer time that is too long. In this case, there is no point in keeping all the solutions that exceed the total allowed transfer time. Thus, at any level, the solutions should be pruned according to their value of Δv and transfer time. The search, on the other hand, is performed trying to achieve solutions with minimum Δv.

In addition, in the real case, the best transfer is not only minimising the Δv: very often, other parameters shall be taken into account, like the total time of flight, the relative velocity with respect to the arrival planet, the inclination, the radius of pericentre and the period of the final orbit, and so on.

For these reasons, a more general framework has been developed. Some assumptions are released, in order to make the method more versatile. Even if Bellman's principle of optimality might not rigorously hold anymore, it will be shown that the method is still capable of providing good results.

First of all, the requirements in Eq. (3.3) and (3.4) are relaxed, i.e. the objective function does not need to be the sum of non-negative terms. Then, the feasible set is defined by a generalisation of Eq. (3.6), as:

$$\overline{D}_i = \left\{ \mathbf{x}_i \in D_i \mid \Phi_i\left(\mathbf{x}_i\right) \right\} \tag{3.10}$$

where the pruning criterion $\Phi_i\left(\mathbf{x}_i\right)$ is a property common to all the solutions in the set. Definition (3.10) includes the ideal case when f and f_i have property (3.3) and $\Phi_i \equiv f_i < \overline{f}_i$.

3.2.4 Application to MGA

The procedure presented in the previous section can be easily applied to the velocity formulation of the MGA trajectory problem, in the way that is presented here.

Let us consider an interplanetary transfer with n_{legs} interplanetary legs, and with a given sequence of planetary swing-bys, starting with a launch. If the velocity formulation is used, then the solution vector associated to the trajectory is the one shown in Eq. (2.17).

$$\begin{aligned}\mathbf{x} = \big[& t_0, v_0, \overline{\theta}, \overline{\delta}, \alpha_1, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, ..., \\ & \gamma_i, r_{p,i}, T_{i+1}, \alpha_{i+1}, ..., \gamma_{n_{legs}-1}, r_{p,n_{legs}-1}, \alpha_{n_{legs}}, T_{n_{legs}} \big]\end{aligned}$$

Also let us consider the problem, very common in preliminary mission design, of minimising the total change in velocity of the spacecraft, provided by the high-thrust engine. Using the model proposed above, in any of the transfer cases, the design of a multi-gravity assist transfer can be transcribed into a general nonlinear programming problem, with simple box constraints. Since, according to the model, the engine is used at launch, in each DSM, and possibly at the final orbit insertion (or rendezvous), then the objective of this problem is to find:

$$\min \left[f\left(\mathbf{x}\right) = v_0 + \sum_{k=1}^{n_{legs}} \Delta v_k + v_f \right] \tag{3.11}$$

The generic Δv_i in Eq. (3.11) can be computed once the trajectory is completed up to leg *i*. This means that only the part of the solution vector $\mathbf{x}$ concerning legs 1 to *i* is needed, and the same value is independent of the variables associated to legs $i+1$ to n_{legs}. This allows splitting the problem into levels: level 1 refers to the part of trajectory from first planet to the second planet; each of the following levels takes into account a swing-by and the subsequent leg – including a DSM – to reach the next planet. So the subdivision of the domain into levels, as shown in Table 3.1 can be considered. The variables related to each level are different, depending on how the trajectory starts, remembering the velocity formulation.

Table 3.1. Levels and related variables.

Level	Variables			Domain
	Initial free launch	Initial ballistic arc	Initial swing-by	
1	v_0, t_0, $\overline{\theta}$, $\overline{\delta}$, α_1, T_1	t_0, T_1	γ_1, $r_{p,1}$, α_1, T_1	$D_{L,1}$
2	γ_1, $r_{p,1}$, α_2, T_2	γ_1, $r_{p,1}$, α_2, T_2	γ_2, $r_{p,2}$, α_2, T_2	$D_{L,2}$
…	…	…	…	…
i	γ_{i-1}, $r_{p,i-1}$, α_i, T_i	γ_{i-1}, $r_{p,i-1}$, α_i, T_i	γ_i, $r_{p,i}$, α_i, T_i	$D_{L,i}$

Physically, this subdivision corresponds to breaking the trajectory into legs, and at each level, to consider the entire trajectory up to the leg corresponding to the level under exam.

This division into levels is arbitrary, but it allows having one (and only one) manoeuvre for each level, regardless the type of transfer (initial free launch, initial ballistic arc, initial swing-by). Generally, this is the DSM. In the specific case of transfer with initial ballistic arc, there is no DSM on the first leg, but the launch excess velocity plays the same role. The reason for this choice is that the manoeuvre magnitude will be one of the main criteria which will be used to prune the solution space.

Due to the objective function chosen in Eq. (3.11), it is straightforward to choose, as partial objective functions, one of the following:

$$\begin{aligned} f_i &= \sum_{k=1}^{i} \Delta v_k \\ f_i &= \Delta v_i \end{aligned} \qquad i = 1,\ldots,n_{legs}-1 \tag{3.12}$$

and since the change in velocity is intended in magnitude, it results that the condition (3.4) on the positivity is satisfied. Without any substantial change, it is also possible to optionally include the terms due to the launch and the final insertion.

Note that the evaluation of the partial objective functions (3.12) requires the computation of i Lambert arcs and i analytical propagations. Therefore, the evaluation of a trajectory (or specifically its Δv, or other cost functions) is computationally cheaper than evaluating the whole solution.

The total amount of Δv provided by the engine in each manoeuvre, and during each phase of the mission, is usually limited by constraints on the engine and availability of fuel. Then, it is the first choice to define, as a pruning criterion for each level, an upper bound on the Δv. This condition allows to prune all the solutions that exceed that amount of change in velocity, even at early legs, since they are not feasible. Obviously this bound is not clearly defined at early stages of mission design, but as it will be shown later in this work, very conservative (high) thresholds are enough to prune substantially the search space.

Again, this has a direct physical meaning: the incremental pruning starts by considering one leg only of the trajectory, from launch to the first planetary swing-by. For this leg, all trajectories are discarded when the change in velocity required to meet the following planet is too high. Certainly, if the first leg of the trajectory is

unfeasible, then it does not make sense to continue: thus the idea of discarding all the trajectories which start with an unfeasible leg. This process corresponds to pruning the solution space of the first level, and not consider it anymore for the following levels. The second level is ideally composed by the first swing-by of the sequence, together with the second interplanetary leg. The second DSM is added to the partial objective function, which is used to prune the search space. The process continues, by incrementally adding legs to the trajectory, and removing unfeasible parts of the domain (hence the name of incremental pruning).

The subdivision into levels proposed in Table 3.1 is certainly not the only possible one. Note that each swing-by is an event on its own, and can be computed, giving the initial conditions, through the two variables r_p, γ. If each swing-by and each deep space flight leg are considered as independent levels, then the subdivision shown in Table 3.2 can be adopted. Since the swing-by is unpowered in this model, no value of Δv is added at the levels of swing-bys. Therefore, if some pruning is to be achieved, different pruning functions, other than an upper bound on the Δv, are to be used.

Table 3.2. Levels, variables and corresponding domains, when considering each deep space flight leg and each swing-by as different levels.

Level	Variables	Domain
1	t_0, θ, δ, α_1, T_1	$D_{L,1}$
2	γ_1, $r_{p,1}$	$D_{L,2}$
3	α_2, T_2	$D_{L,3}$
4	γ_2, $r_{p,2}$	$D_{L,4}$
…	…	…
$i-1$	γ_{i-1}, $r_{p,i-1}$	$D_{L,i-1}$
i	α_i, T_i	$D_{L,i}$

3.2.5 Discussion

Before proceeding to the next section, it is worthwhile to examine some of the characteristics of the proposed incremental approach.

One key assumption of the incremental approach is that a complete solution to the MGA problem, i.e. a complete trajectory, can be built by adding individual trajectory legs, starting from departure to the arrival or vice versa. Therefore, although the global minimum of each sub-problem does not represent the global minimum of the whole problem, we can build the solution space of the whole problem by incrementally adding up the search spaces associated to each sub-problem in such a way that the resulting total search space contains the global minimum.

The objective functions that are used to prune the search space associated to each sub-problem do not directly depend on the chosen objective function for the whole problem. Therefore, the incremental approach is independent of the objective

function of the whole problem, but is strongly dependent on the characteristics of the trajectory model.

In particular, for the trajectory model in velocity formulation, which will be used in the following, the partial objective function (and pruning criterion) associated to each sub-problem cannot be evaluated without considering all the previous levels. This represents a fundamental difference with respect to what was done in [59], by using the position formulation of the MGA problem. In fact, a trajectory model in which gravity manoeuvres are modelled as powered swing-bys does not need to build the whole solution incrementally (or as a cascade of sub-problems) but each sub-problem can be tackled in parallel with the others. Furthermore, in the proposed incremental approach, the search space is constructed incrementally, therefore the number of dimensions of each sub-problem increases as a new level is added to the list. On the other hand, the number of dimensions of each sub-problem in [59] remains constant throughout the whole pruning process.

Furthermore, conceptually, this approach can be equally applied to the position formulation without modification if we assume that the feasible set is made of those solutions that satisfy the constraints and have f_i is below a given threshold. It can also be applied to the block approach, without any substantial modification.

3.3 Incremental Pruning with Systematic Search on a Grid

For the purpose of showing how the incremental pruning process works, at first a very simple implementation of it will be introduced. As a search method for defining the feasible sets at each level, a systematic grid search will be used. Although quite inefficient, this method is ideal to show the effectiveness of the incremental pruning, for at least two different reasons. Firstly, it is fully deterministic, thus repeated runs of the algorithm provide identical results. Secondly, the grid search guarantees to find all the existing solutions in the domain, as long as the grid is fine enough.

In order to perform the grid search, the intervals defining the domain D on each coordinate $\left[x_l \; x_u\right]_i$, for $i = 1,...,n$ are divided into p_i sub-intervals: this means that the whole domain of D is divided into $\prod_{i=1}^{n} p_i$ hyper-rectangles. Let us call each one of these hyper-rectangles a *node* (Fig. 3.2). Modern techniques exist to estimate the minimum value of a function given some sample points, for example by building surrogate models of the function [109]. These methods go beyond the scope of this test case. In fact, if we simply assume that the mesh is fine enough, together with a finite value of the Lipschitz constant, then it is implied that one sample in each node (i.e. the middle point) is a good approximation of the value of f_i.

After the discretisation, the feasible set, at each level, becomes a set of nodes, which can be enumerated. If no pruning is performed at a generic level *i*, i.e. all the

nodes are considered, the total number of nodes is $\prod_{l=1}^{|D_l|} p_l$, where $|\cdot|$ is the dimensionality of the domain. Starting from the second level, and considering that the level $i-1$ has been pruned, the number of nodes at level *i* reduces to:

$$n_{nodes,i} = \overline{n}_{nodes,i-1} \prod_{l=|D_{i-1}|+1}^{|D_i|} p_l, \qquad i \geq 2 \tag{3.13}$$

where $\overline{n}_{nodes,i-1}$ is the number of nodes that survived after pruning level $i-1$. Since, due to the pruning, it results that:

$$\overline{n}_{nodes,i-1} \leq \prod_{l=1}^{|D_{i-1}|} p_l, \qquad i \geq 2,$$

then pruning reduces the number of nodes to be considered at the following level. If the pruning at level $i-1$ is effective, the number of evaluations at level *i* can be substantially reduced.

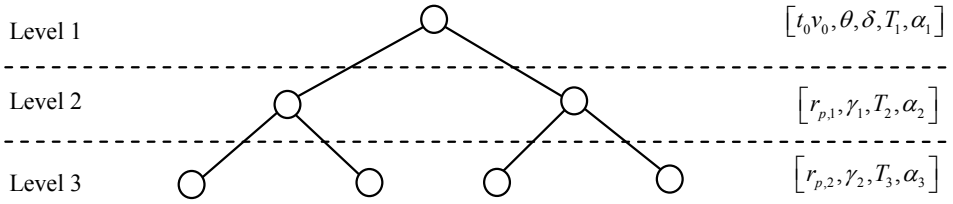


Fig. 3.2. Tree representation of a MGA trajectory: each node is a stage composing the trajectory.

3.3.1 Test Case

Consider a transfer from Earth to Jupiter via Venus, Earth, Earth swing-bys. We consider no DSM in the first leg (and thus the first leg is a ballistic arc), and the subdivision into levels shown in Table 3.2.

The domain bounds and the number of intervals for each variable are shown in Table 3.3. These bounds have been chosen because from [41] it is know that they define a domain *D* that contains an optimal solution.

The hyperbola pericentre radius r_p is normalised with respect to the radius of the planet at which the swing-by is performed, and the intervals for this variable are not equally spaced over the domain, but they are smaller for low values of the radius. This is done because the outgoing velocity from a swing-by is much more sensitive to the variation of the radius, when the radius is small, than when it is large. This choice allowed the use of fewer intervals for the variable r_p.

Table 3.3. Domain bounds and number of intervals for each variable.

Level	Variable	Lower bound	Upper bound	No. of intervals (p_i)
1	t_0 , MJD2000	3159	3559	19
	T_1 , d	120.4	220.4	19
2	γ_2 , rad	2.742	3.789	19
	$r_{p,2}$	1.672	2.272	9
3	α_2	0.01197	0.2619	19
	T_2 , d	220.2	420.2	19
4	γ_3 , rad	2.127	3.174	19
	$r_{p,3}$	1.466	2.066	9
5	α_3	0.01728	0.2772	19
	T_3 , d	630.4	830.4	19
6	γ_4 , rad	2.638	3.685	19
	$r_{p,4}$	1.312	1.912	9
7	α_4	0.01740	0.3074	19
	T_4 , d	747.0	947.0	19

Considering the first level of the problem only, it is possible to plot what is commonly called *pork chop plot*, that is a plot of the Δv needed at the first planet (in this case, Earth) in order to reach the second planed (Venus) as a function of the starting date t_0 and the time of flight T_1 (Fig. 3.3). The grid in the plot is representative of the intervals in which the domain has been divided into: each hyper-rectangle of the grid is a node, and it is represented by its middle point. In this example, the function has been evaluated once, for each node, in the centre of the corresponding sub-domain.

If the solution space of the first level is pruned as shown before, setting a pruning criterion:

$$\Phi_1 = \left(\Delta v_1 < 5 \text{ km/s}\right),$$

the algorithm removes all the parts of this domain which exceed this limit. The only remaining areas are shown in Fig. 3.4.

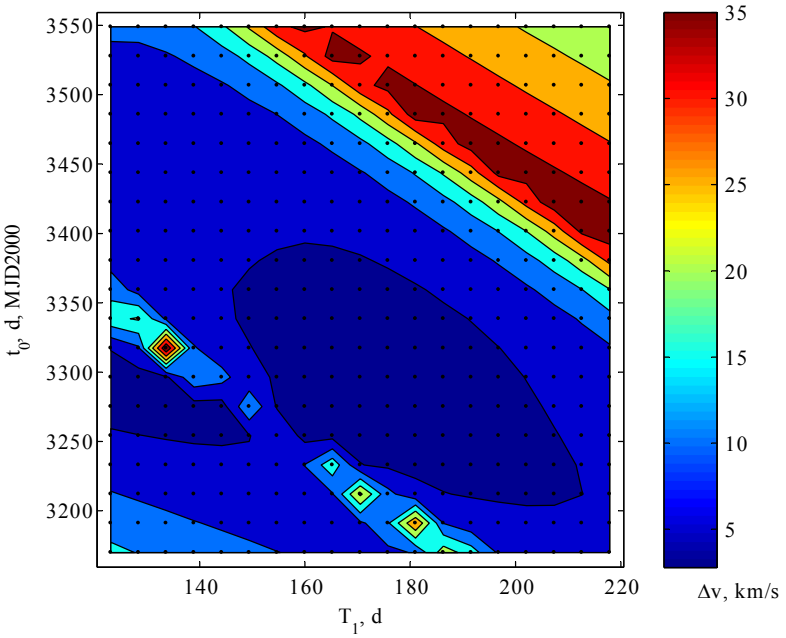


Fig. 3.3. **Level 1, no pruning done. Transfer cost (Δv) as a function of departure time t_0 and time of flight T_1. The black dots represent the grid sample points.**

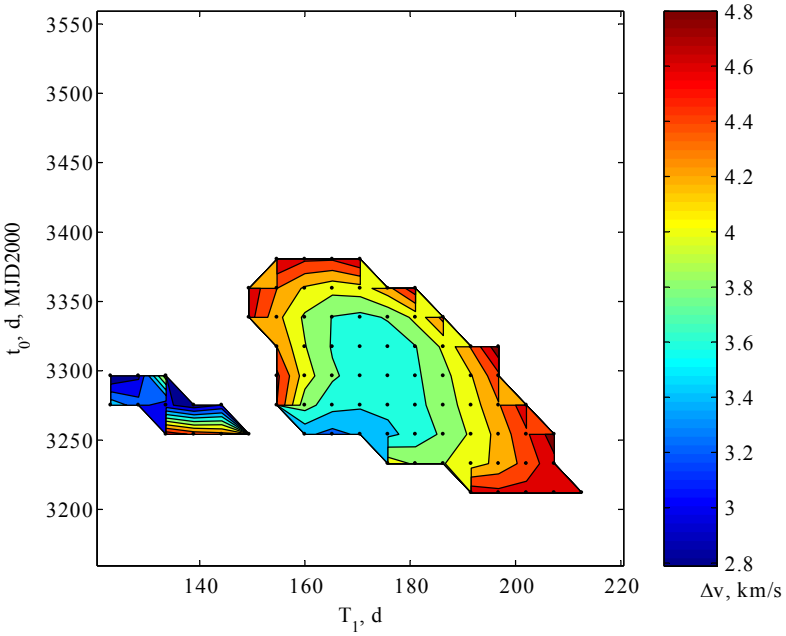


Fig. 3.4. **Level 1, after pruning. Transfer cost (Δv) as a function of departure time t_0 and time of flight T_1. Only feasible nodes have been plotted.**

The coloured areas in Fig. 3.4 represent the domain that survived after pruning at first level $\bar{D}_1$. The second level is a swing-by, thus the Δv is not a good pruning criterion. Instead, by considering the aim of this particular swing-by, that is to increase the semimajor axis of the transfer orbit, it is possible to prune the second level by requiring that the Venus swing-by increases the semimajor axis by at least 27,986,077 km. That is:

$$\Phi_2 = \left(\Delta a > 27986077 \text{ km/s}\right)$$

where Δa is the difference between the semimajor axis before and after the swing-by of Venus. This quantity can easily be computed as the swing-by fully determines the conditions of the outgoing heliocentric orbit. The numerical value for pruning has been set from the various data in literature about similar trajectories [41, 74]. Then, since in the third level there is an impulsive manoeuvre, the criterion is a limit on Δv_2:

$$\Phi_3 = \left(\Delta v_2 < 0.5 \text{ km/s}\right).$$

The value is suitable for a DSM. The results of the pruning up to level 3, with these criteria, are shown in Fig. 3.5 to Fig. 3.7. Fig. 3.5 shows the pork chop plot of the first two variables. The number of feasible nodes is clearly decreased with respect to Fig. 3.4. This is due to the fact that the pruning of levels 2 and 3 has made part of the nodes at level 1 infeasible. The infeasible nodes are then removed from level 1 according to the back pruning strategy explained in Section 3.2.2.

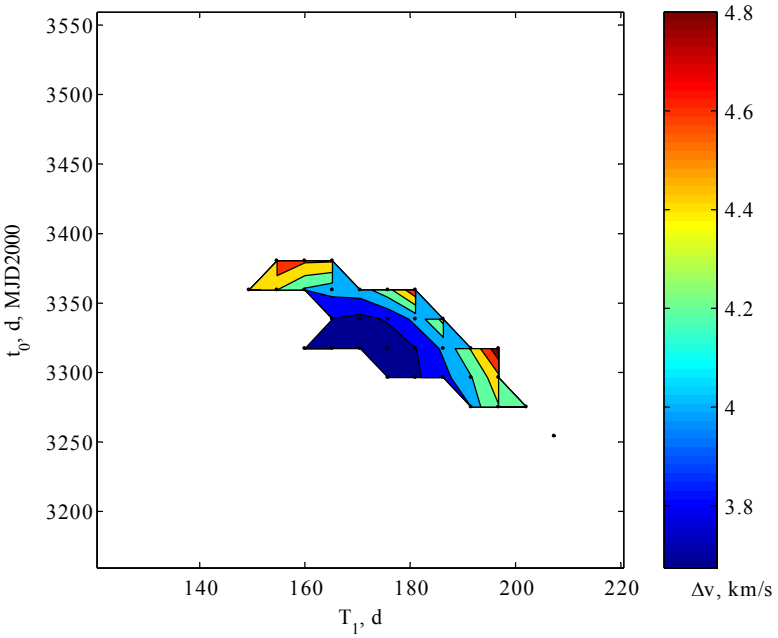


Fig. 3.5. Level 1, after pruning up to level 3. Transfer cost (Δv) as a function of departure time and time of flight. Only feasible nodes have been plotted.

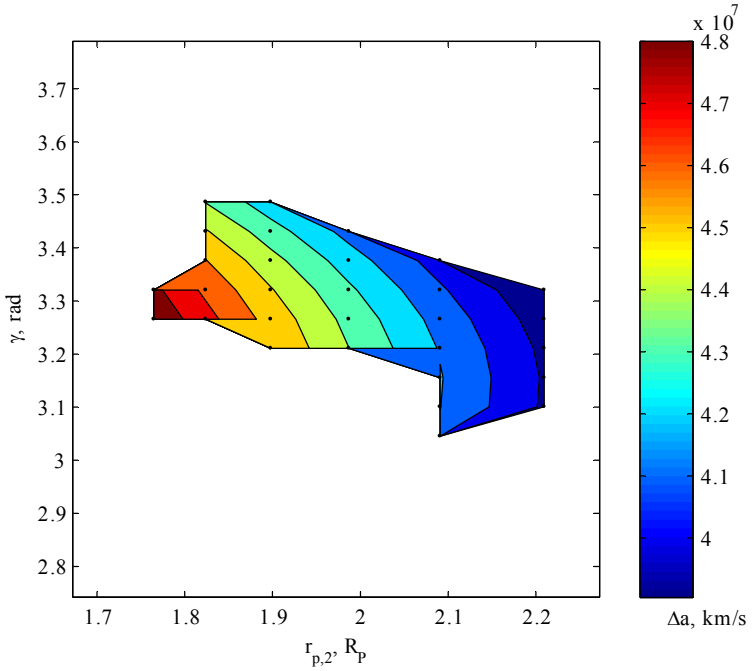


Fig. 3.6. **Level 2, after pruning up to level 3. Increment of semimajor axis (Δa) due to the swing-by of Venus, as a function of the rotation angle and the hyperbola pericentre radius. Only feasible nodes have been plotted.**

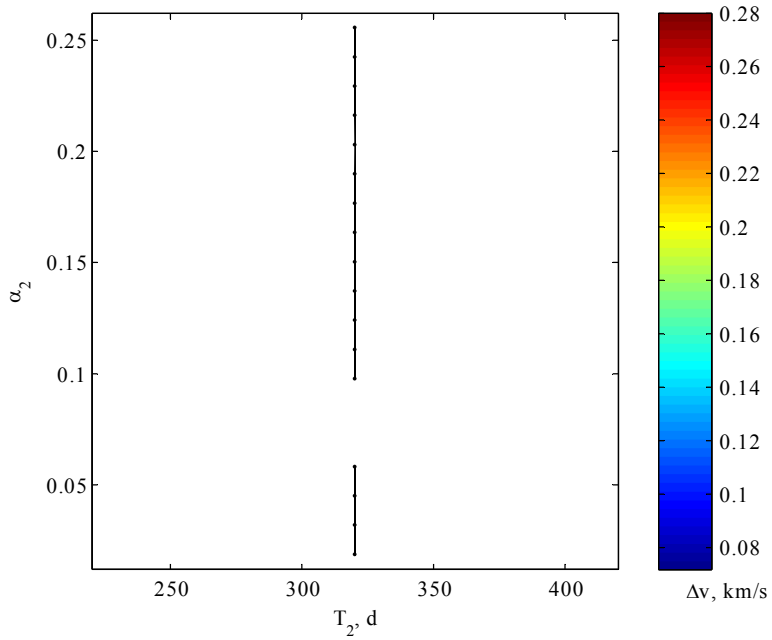


Fig. 3.7. **Level 3, after pruning up to level 3. Cost of the deep space manoeuvre (Δv) as a function of the DSM position α_2 and time of flight T_2. Only feasible nodes are plotted.**

In Fig. 3.6, the increment of the semimajor axis due to the Venus fly-by is represented, as a function of the variables of the second level, and fixed the values of the first level, as $t_0 = 3359$ d, MJD2000 , $T_1 = 170.5$ d .

By fixing also the values of the variables of the second level ($\gamma_2 = 3.265$ rad , $r_{p,2} = 2.090$), the Δv_2 of the third level can be plot as a function of α_2 and T_2 (Fig. 3.7). The plot highlights that only nodes with a T_2 of about 320 days have survived the pruning process. Furthermore, the timing of the DSM is not particularly important for pruning the solution space, as almost any value of α_2 within the considered global bounds generates at least one solution, which satisfies all pruning criteria.

The whole search space can be pruned down, until to the last, level by continuing this process of incremental and backward pruning. Fig. 3.8 represents one of the possible feasible trajectories. The solution in Fig. 3.8 corresponds to the following solution vector:

$$\begin{aligned}\mathbf{x} = \big[&3359, 170.5,\\ &3.265, 2.090, 0.05802, 320.2,\\ &2.651, 1.766, 0.06517, 730.4,\\ &3.162, 1.612, 0.02503, 941.8\big]\end{aligned}$$

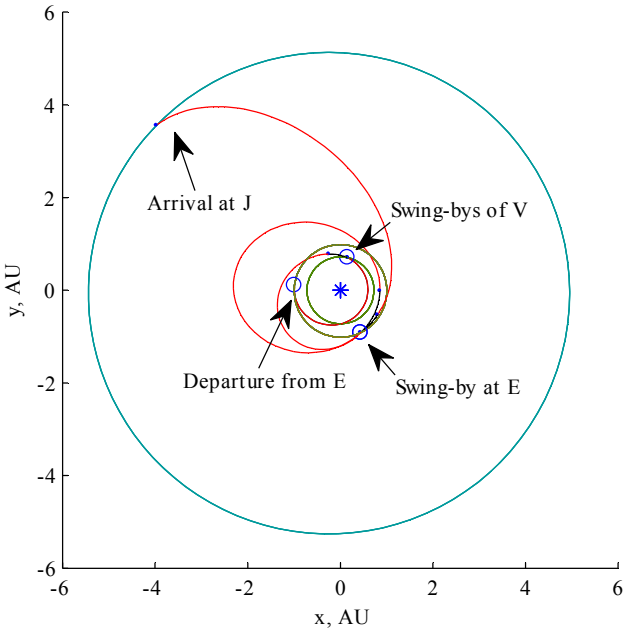


Fig. 3.8. One of the feasible trajectories after the pruning of the whole domain.

The total Δv of this trajectory is about 5.34 km/s, without considering any capture manoeuvre at Jupiter. As a comparison, the best known solution, for the specified bounds, requires 4.84 km/s. It is worth to note that no search for locally optimal solution was performed: the solution in Fig. 3.8 was found only as the result of the systematic search over all the nodes survived to the pruning process.

3.3.2 Algorithmic Complexity Analysis

The computational complexity of the incremental pruning method presented above is dictated by the number of nodes that need to be evaluated. If no pruning is performed at the first level then all the nodes have to be evaluated. The total number of evaluations is simply:

$$n_{nodes,1} = \prod_{l=1}^{|D_1|} p_l$$

If pruning is applied the number reduces to $\overline{n}_{nodes,1} < n_{nodes,1}$. The number of function evaluations at the following level $i+1$ is computed recursively through Eq. (3.13) as a function of the remaining nodes at level *i*. This implies that, depending on the efficiency of the pruning, the total number of function evaluations used by the pruning process could be higher than the number of function evaluations needed to scan the whole grid of the complete problem, that is:

$$n_{nodes,n_{legs}} = \prod_{i=1}^{n} p_i \tag{3.14}$$

In fact, if for example we consider the extreme case when no node is removed at any level, then the total number of function evaluations through the incremental approach is

$$\prod_{l=1}^{|D_1|} p_l + \prod_{l=1}^{|D_2|} p_l + \ldots + \prod_{l=1}^{n} p_l$$

that is certainly bigger than the number of function evaluations needed if a systematic scan is performed on the whole problem (Eq. (3.14)).

Again it is worthwhile reminding that even if the number of function evaluations in the incremental pruning is higher, still the total computational time needed by the process can be lower, due to the fact that function evaluations at early levels are less expensive. Of course, at this stage and due to the number of parameters involved in the problem, it is difficult to determine if and when the incremental pruning process is beneficial. Specific tests in following sections will show the benefit of the incremental pruning.

3.4 Identification of the Feasible Set

In Section 3.3, the search for the feasible set was performed with a systematic search. Although this method presents some advantages (namely determinism and completeness of the search within grid accuracy), its algorithmic complexity is exponential with problem dimensions. Furthermore, if a coarse grid and a bland pruning are used, many optimal solutions are likely to be lost; on the other hand, if a sufficiently fine grid is used, together with an aggressive pruning, the computational time becomes quickly unacceptable even for a limited number of planetary swing-bys.

A solution to this problem is to use a search method that uses some heuristics to improve the search, rather than systematically sampling all the points on a grid. The search method can also assume some properties of the function, like continuity or differentiability, to compute the best search direction. In the following, a number of optimisers for single objective optimisation will be used to search and identify the feasible sets.

Although dropping the grid search removes an exponentially complex step in the search for the feasible set, it also removes the exhaustiveness of the search. Common algorithms for single objective optimisation, in fact, identify a limited number of feasible local minima. These points belong to the feasible set, but do not necessarily characterise it completely. Therefore, there is the need for converting a finite set of feasible points (provided by the optimiser) into a feasible region (that can be used in the incremental process).

A further difficulty in the identification, and use, of the feasible set, at a given level, is its shape. In fact, the representation and storage of an n-dimensional disconnected set of arbitrary shape is a very demanding process which would increase the computational load and the memory storage of the residual space.

Therefore, it was decided to define all the feasible sets as a collection of hyper-rectangles in the search space. The hyper-rectangles, or *boxes* for simplicity, have their edges aligned with the axes of the domain, such that simple box constraints can be used to define each box. Furthermore, the boxes are not necessarily part of a regular grid, but their position and size is in principle free in the search space (to achieve a tight enclosure of the feasible region).

In the following sections, different methods to define the feasible set at each level will be explained. In particular, one section will describe the optimisers used to search for feasible points in the search space, then a section will follow with different methods for clustering the solution into boxes. These two parts of the process are distinct, as in principle any optimiser and any box creation method can be used for a given problem.

3.4.1 Exploration of the Search Space

In this section, the search methods used to identify feasible points will be illustrated. We assume that the search is performed on a box-constrained search space. This assumption is consistent with all the techniques, which will be used to generate the feasible set. The search methods will be applied either to a single box

containing a subset of the whole search space, or to a collection of all the boxes containing feasible regions.

MULTI-START SEARCH

A very simple global optimisation procedure can be created by iterating several local minimisations in different points of the search space. This technique is known as Multi-Start (MS) search. There exist a number of variations of this method, including different heuristics for the selection of the starting points, and different algorithms for the local search [110].

For this work, a simple implementation of a Multi-Start technique was used. Its simplicity may cause some concerns, as nowadays powerful advanced global optimisers exist, but the point here is to prove the effectiveness of the incremental pruning method, and not strictly the optimiser itself. Furthermore, the simplicity of the optimiser itself allows us to focus on the incremental method, rather than on tuning the many parameters of other more complex optimisers.

Each Multi-Start-based technique has two phases: the first one, which can be considered a global search, is determining the starting points for the subsequent local optimisations, which will be performed in the second phase. If no additional knowledge of the domain is available, the starting points (or samples) are distributed uniformly in the whole search space. The uniform distribution is a very common choice among all the population based optimisers. The number of samples used, n, is a key parameter of the optimisation process. The objective function is then evaluated in each one of the samples. The samples are sorted according to the value of the objective. The best $m < n$ samples are selected and used as a first guess for starting m local optimisations. This second step has the task of finding the local minimum in the closest basin of attraction, and a gradient-based method is generally used. In this work, the function *fmincon* (which is an implementation of Sequential Quadratic Programming, SQP) in the Optimization Toolbox of MATLAB® is utilised.

Gradient based methods require the differentiability of the objective function, which is not always the case for the applications that we will present, in particular when a space transformation is applied. We can still assume that the function is generally differentiable and discontinuities are limited and can be avoided by random sampling.

The local optimisation process is an iterative algorithm that tries to reach the bottom of the basin of attraction of a local minimum. The solution moves iteratively towards the minimum, having a lower objective value at each iteration. Since the aim of the search is not to find the local minimum, but to find the feasible set, we are not interested in fully converging. Instead, we can stop the optimisation as soon as the current point has a function value that is lower than a given threshold. The motivation to this can be found very easily remembering the ideal case presented in Section 3.2.4. If the partial objective function is a sum of Δv, and the pruning criterion is an upper bound of this sum, the condition mentioned before is equivalent to stopping the optimisation as soon as a feasible point is found. This trick has two benefits: the first is that it saves a large number of function evaluations: in fact, it is generally true that a local optimiser spends a lot of

computational power to refine to high precision the position of the minimum, but for the pruning, we are not interested in that. The second advantage is that it allows having feasible points which are sparse in the feasible set, without accumulating at the local minimum. This is necessary to identify correctly the whole feasible set.

MODIFIED MONOTONIC BASIN HOPPING

A more sophisticated search method can be derived from Monotonic Basin Hopping. MBH was first applied to special global optimisation problems, the molecular conformation ones [111], and later extended to general global optimisation problems [112]. In its basic version it is quite similar to the Multi-Start. It is also based on multiple local searches and the only difference is represented by the distribution of the starting points for local searches: while in Multi-Start these are randomly generated over the whole feasible region, in MBH they are generated in a neighbourhood $N\left(\mathbf{x}^*\right)$ of the current local minimiser $\mathbf{x}^*$. The modified MBH is described in Algorithm 3.1.

Algorithm 3.1. Modified MBH.

1: Select $\mathbf{x}$ in D_i, initialize $n_{eval} = 0, n_{trials} = 0$
2: Run local optimiser from $\mathbf{x}$ to local minimum $\mathbf{x}^*$
3: Select a candidate point $\mathbf{x}_c \in N\left(\mathbf{x}^*, \rho_l\right)$;
Update n_{eval}; $n_{trials} \leftarrow n_{trials} + 1$
4: **If** $n_{trials} > n_{trials,max}$
5: goto Step 1
6: **End If**
7: **If** $\Phi_i(\mathbf{x}_i)$ **Then**
8: $L_{feas} \leftarrow L_{feas} \cup \{\mathbf{x}_c\}$; goto Step 3
9: **End If**
10: Run local optimiser from $\mathbf{x}_c \rightarrow \mathbf{x}^*_{new}$, update n_{eval}
11: **If** $f_i\left(\mathbf{x}^*_{new}\right) \le f_i\left(\mathbf{x}^*\right)$ **Then**
12: $\mathbf{x}^* \leftarrow \mathbf{x}^*_{new}$
13: **If** $\Phi_i(\mathbf{x}_i)$
14: $L_{feas} \leftarrow L_{feas} \cup \{\mathbf{x}^*\}$
15: **End If**
16: **If** $f_i\left(\mathbf{x}^*_{new}\right) < f_i\left(\mathbf{x}^*\right)$
17: $n_{trials} = 0$
18: **End If**
19: **End If**
20: **Termination** Unless $n_{eval} \ge n_{max}$, goto Step 3

More in detail, given a local minimum $\mathbf{x}^*$ and a neighbourhood of this local minimum $N\left(\mathbf{x}^*,\rho_l\right)\subseteq D_i$, with radius ρ_l, MBH selects a random point $\mathbf{x}_c\in N\left(\mathbf{x}^*,\rho_l\right)$ and runs a local optimisation. If the new local minimum $\mathbf{x}^*_{new}$ obtained starting from the candidate point $\mathbf{x}_c$ is better (i.e. lower value of the objective function) than $\mathbf{x}^*$, then $\mathbf{x}^*=\mathbf{x}^*_{new}$. MBH saves only the local best and therefore would be unusable to explore the feasible set once one point is identified. Therefore, MBH was modified to store all candidate local optimal points $\mathbf{x}_c$ or $\mathbf{x}^*_{new}$ that satisfy the conditions $\Phi_i(\mathbf{x}_i)$.

The modified MBH was complemented with a restart of the process after a number $n_{trials,max}$ of local trials. This restart is fundamental to avoid stagnation and coverage only of one portion of the feasible set when the feasible set is disconnected.

MODIFIED MULTI-AGENT COLLABORATIVE SEARCH

The third technique used in this thesis is an evolutionary based one derived from the Multi-Agent Collaborative Search (MACS) described in [66]. Unlike MS and MBH, it is derivative free. The basic idea underneath the Multi-Agent Collaborative Search is to assign the task of looking for a set of solutions to a population *P* of agents that performs a combination of local and global searches. An agent is identified by a solution vector $\mathbf{x}_j$, a sub-region of the search space $N\left(\mathbf{x}_j,\rho_j\right)\subseteq D_i$, and a local search operator, or individualistic behaviour function $B\left(\mathbf{x}_j,n_{trials}\right)$ that generates n_{trials} samples $\mathbf{x}_{j,l}$ such that $\mathbf{x}_{j,l}\in N\left(\mathbf{x}_j,\rho_j\right)$. At every generation *g* a communication operator recombines pair-wise the agents in the current population P_g and generates a new candidate population P_c made of solution points $\mathbf{x}_{j,c}$. A greedy selection operator selects only the solutions in P_c that improve the solutions in P_g and updates P_g. Before applying the communication operator, the points in P_g that satisfy condition $\Phi_i(\mathbf{x}_i)$ are stored in an archive L_{feas}. After the communication operator, the local search operator is applied to the best $n_{popratio}<n_{pop}$ agents. All the sample points $\mathbf{x}_{j,l}\in N\left(\mathbf{x}_j,\rho_j\right)$ satisfying condition $\Phi_i(\mathbf{x}_i)$ are stored in the archive L_{feas}. The overall MACS is presented in Algorithm 3.2.

Furthermore, in order to facilitate the local exploration in a neighbourhood of a feasible solution without any modification to the two search algorithms, we assigned the value -2 to the objective function of all the feasible solutions.

Algorithm 3.2. Modified MACS.

1: Initialize a population P_0 in D_i, with n_{pop} agents, initialize $n_{eval} = 0, g = 0$
2: $\forall \mathbf{x}_j \in P_g$, **If** $\Phi_i(\mathbf{x}_i)$ **Then**
3: $L_{feas} \leftarrow L_{feas} \cup \{\mathbf{x}_j\}$
4: **End If**
5: Apply communication operator: $com: P_g \rightarrow P_c$; $n_{eval} \leftarrow n_{eval} + n_{pop}$
6: Apply greedy selection operator:
$\forall \mathbf{x}_j \in P_g, \mathbf{x}_j \leftarrow \mathbf{x}_{j,c} \in P_c$ if $f(\mathbf{x}_{j,c}) \le f(\mathbf{x}_j)$, for $j = 1,...,n_{pop}$
7: Rank the population P_g
8: Apply local search operator to the best $n_{popratio}$ agents in P_g:
$\forall \mathbf{x}_j \in P_g$ generate n_{trials} candidate points $\mathbf{x}_{j,l} \in N(\mathbf{x}_j, \rho_j)$;
$n_{eval} \leftarrow n_{eval} + n_{popratio} n_{trials}$
9: Apply greedy selection operator:
$\mathbf{x}_j \leftarrow \mathbf{x}_{j,l}$, for $j = 1,...,n_{popratio}$ and $l = 1,...,n_{trials}$, if $f_i(\mathbf{x}_{j,l}) \le f_i(\mathbf{x}_j)$
10: **If** $\Phi_i(\mathbf{x}_i)$ **Then**
11: $L_{feas} \leftarrow L_{feas} \cup \{\mathbf{x}_{p,l}\}$
12: **End If**
13: $P_{g+1} \leftarrow P_g; g \leftarrow g + 1$
14: **If** $\forall \mathbf{x}_j, \mathbf{x}_k \in P_g, \max_{j,k} \|\mathbf{x}_j - \mathbf{x}_k\| < tol_{conv}$ **Then**
15: Restart the worse of the two
16: **End If**
17: **Termination** Unless $n_{eval} \ge n_{eval,max}$, goto Step 5

For MACS we adopted a restart technique similar to modified MBH, but the domain D_i is partitioned in sub-domains at every restart and MACS is restarted within a sub-domain. The domain D_i is partitioned by dividing in two parts one coordinate belonging to a subset of the coordinates at level *i* (for example only the second and the third coordinate are divided while the others remain unchanged). At each restart, the coordinate with the longest edge is cut in two parts generating two new sub-domains. For each sub-domain we evaluate the two criterion vector:

$$\boldsymbol{\psi}_{D_q} = \left[-V_{D_q}, \frac{n_{feas,D_q}}{n_{feas,D_i}} \right] \tag{3.15}$$

where V_{D_q} is the volume of the sub-domain D_q and $n_{feas,D_q} / n_{feas,D_i}$ is the ratio between the feasible solutions n_{feas,D_q} in $D_q \subseteq D_i$, such that $D_i = \bigcup_q D_q$, and the

total number of feasible solutions n_{feas,D_i} in D_i. For each sub-domain we evaluate its Pareto optimality with respect to the criteria vector in Eq. (3.15) by computing the dominance index:

$$I_d\left(D_q\right) = \left|\left\{ j \mid D_j \succ D_q \right\}\right| \tag{3.16}$$

where $|.|$ is the cardinality of the set and a sub-domain dominates another when both criteria are better, i.e. $D_j \succ D_q \Rightarrow -V_{D_j} < -V_{D_q} \wedge n_{D_j}/n_{D_i} < n_{D_q}/n_{D_i}$.

Furthermore, we count the number of subdivisions $p_d\left(D_q\right)$ that do not produce any increase in the number of feasible solutions. For example, assume that the number of feasible solutions in D_i is 100 and that we subdivide D_i in D_q and D_{q+1}; then the subdivision index for each of the sub-domains is increased by one unit. After selecting one of the two, we run MACS, if the number of feasible solutions is higher than 100, then p_d is set to 0. In order to select a sub-domain where we want to restart MACS, we use the cumulative quality index $I_q\left(D_q\right) = I_d\left(D_q\right) + p_d\left(D_q\right)$. If more sub-domains have the same quality index we pick one randomly among them.

3.4.2 Clustering of the Solutions

The search for the feasible solutions at each level ends with a number of points, the objective value of which is below a given threshold (feasible points). The following step is to identify and bound the feasible set, i.e. the regions of the search space that have not been pruned.

This section will show the techniques to generate the feasible set, defined as a set of hyper-boxes, starting from a set of feasible points given by one of the search techniques described above. Four different methods will be presented that were developed and used throughout this thesis. Each one has advantages and drawbacks.

METHOD 1: ENVELOPING BOXES

This first basic method was developed for preliminary tests. The idea behind this method is based on the assumption that the global optimum of the complete problem is in the vicinity of one of the local optima of each level.

Therefore, following this assumption, one box is generated for each feasible point found; the solution is centred in the corresponding box; all the boxes have the same size, which is defined *a priori* by the user. The boxes can go out of the global bounds of the problem. In that case, the box is shrunk to fit inside the domain.

If two feasible points are close each other, then the two corresponding boxes generated from them will be overlapping. In principle, the feasible set does not take into account overlapping, but since the boxes are treated as different spaces in the following levels, then the search space in the overlapped part is searched multiple times. This is not a real issue, as multiple overlapping boxes correspond to multiple

local minima at close distance. Thus, the surrounding search space is worth being investigated. The drawback is that the number of boxes equals the number of feasible solutions, and thus can become very high. In particular, an exhaustive search requires a high number of solutions, but this generates a lot of boxes. The high number of boxes can become problematic for problems with more than 2 levels, due to the number of subsequent optimisations needed. Fig. 3.9 shows an example of boxes generated with this method, given the solutions represented with a red dot.

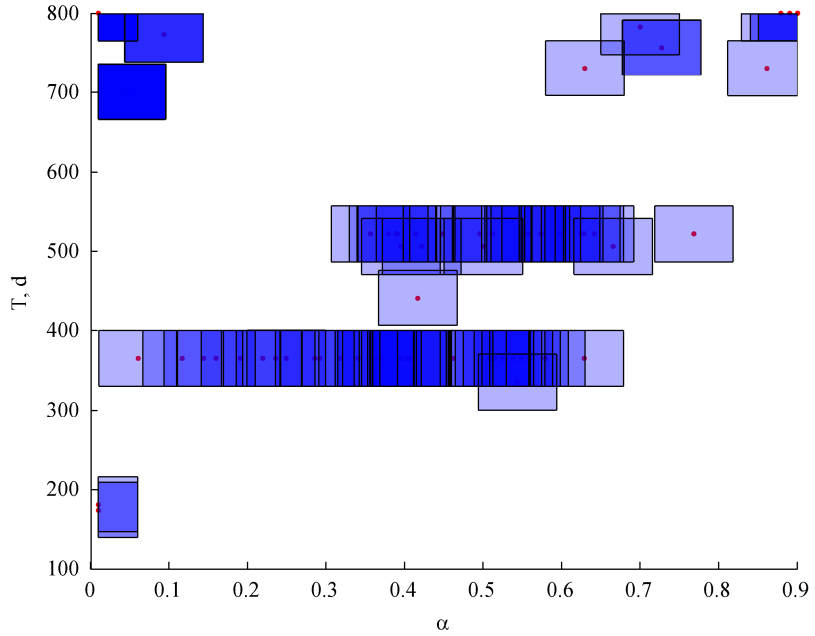


Fig. 3.9. Generation of the boxes by enveloping each solution (method 1). As it can be seen from this example, boxes overlap consistently along the T = 365 d line, where several solutions have been found. Also, solutions at the border of the search space generate smaller boxes, as the boxes cannot exceed the global bounds.

METHOD 2: BOXES ON A GRID

This second method was designed to limit the number of boxes that were created using method 1, i.e. to avoid to create one box for each feasible solution found. In this way, an extensive search, which provides a high number of feasible solutions, can be performed, still keeping the number of resulting boxes reasonable.

Similarly to what happens by using a grid search, all the boxes lay on a grid with pre-defined spacing. Each box is selected if and only if contains a feasible solution. If a box contains more than one solution, it is still considered only once. In this way, overlapping of the boxes is not possible, and the union of the resulting boxes represents the feasible set exactly once, without repetitions.

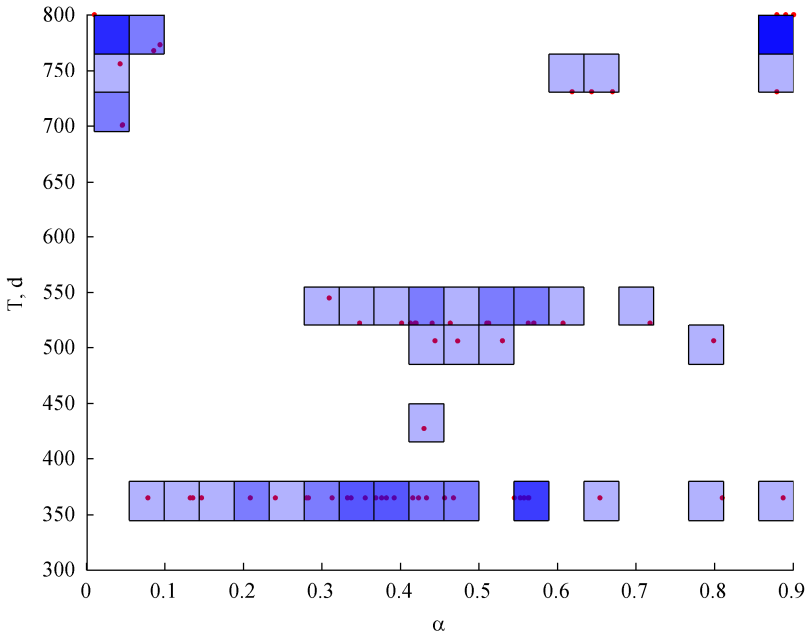
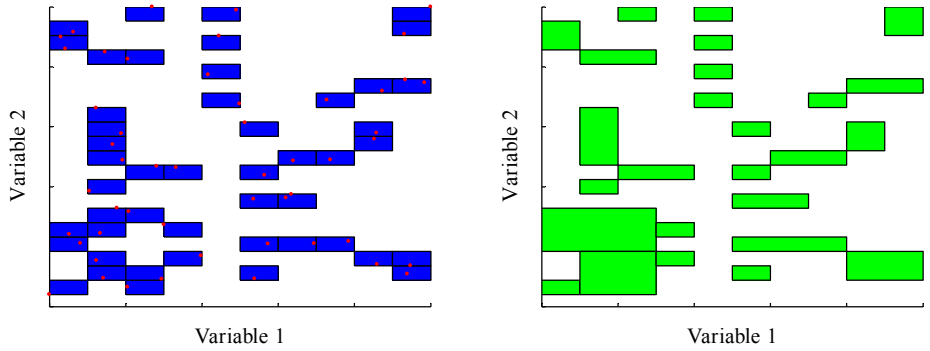


Fig. 3.10. Generation of the boxes by choosing boxes on a grid (method 2). This example of clustering highlights that a coarser grid along dimension α could considerably reduce the number of boxes without increasing the size of the feasible set.

The main problem of this method is once again the resolution of the grid, which has to be chosen a priori. In fact, despite a fine grid is advisable to identify in detail the areas of the feasible set, a large number of boxes will be generated. On the other hand, a coarse grid will result in a not very effective pruning. Fig. 3.10 shows an example of boxes generated with this second method.

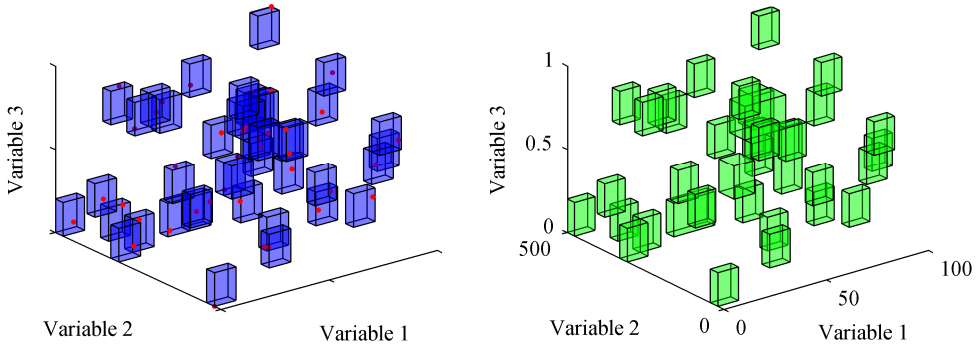
METHOD 3: BOXES ON A GRID AND WRAPPING

This third method is in fact an extension of method 2. The idea to overcome the drawback of high number of boxes generated is to collect within a bigger box all the boxes that share one side in the multi-dimensional space. In this way, all the boxes that are aligned along a given axis, for example as in Fig. 3.10, would become a single box. The aim of the last step is not only to reduce the number of the boxes, but also to envelope in one box the regions in which there are many local minima close one another, preserving the local and adjacent structures of the solution space. Fig. 3.11 and Fig. 3.12 show the last phase of the box creation process, for an ideal two-dimensional and three-dimensional space respectively. In plots (a), the boxes generated by through method 3 are shown, together with the feasible points used as a start. Method 3 reduces the number of boxes by enveloping adjacent ones like in plots (b).



(a) Box creation (b) Box enveloping

Fig. 3.11. The box creation process according to method 4, a two-dimensional space. In (a), the boxes as created by method 3. In (b) is the application of method 4: adjacent boxes are enveloped, and the number of boxes decreases.



(a) Box creation (b) Box enveloping

Fig. 3.12. The box creation process according to method 3, a three-dimensional space. In (a), the boxes as created by method 2. In (b) is the application of method 4: adjacent boxes are enveloped, and the number of boxes decreases.

METHOD 4: MEAN SHIFT CLUSTERING

This last method makes use of a clustering algorithm to identify the sets of points that belong to the same feasible regions in the search space. In the following we will use the Mean Shift clustering algorithm [113]. The Mean Shift clustering takes a number of points as an input in an *n*-dimensional space and returns which point belongs to each cluster. The number of clusters that are generated for a given set of input points is not pre-defined, but is controlled by a single parameter, the bandwidth, that has to be provided by the user. The relative distance among the points determines to which cluster each point belongs to.

It is to underline that the algorithm as presented in the paper exploits a few iterations of a non-deterministic optimisation process, and thus the resulting clustering can differ slightly from run to run.

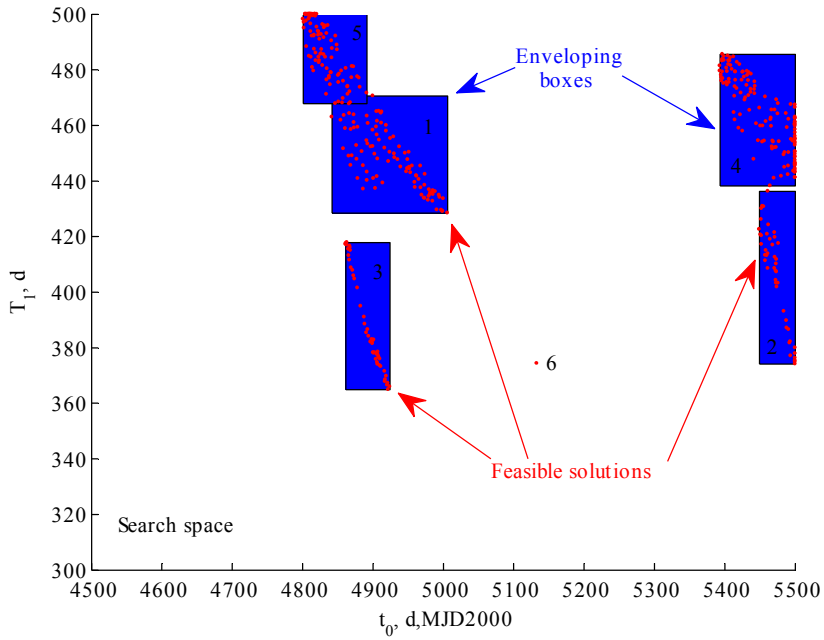


Fig. 3.13. Representation of a two-dimensional search space, with feasible solutions and boxes which envelope the solutions and generate a feasible region.

The Mean Shift clustering method itself does not provide any feasible set, but only clusters for the feasible points. If the search is exhaustive, we can assume that the feasible points are well distributed among the feasible set that we are investigating. Under this assumption, we can state that each cluster identified by the Mean Shift clustering method is a connected feasible region in the search space. At this point it comes again the problem of describing a region given a finite set of points. Some further assumptions can be taken, for example that each region is convex. This will generate a set of convex hulls, each of those can be described with a set of inequalities. Since we would like to define the feasible space as a set of hyper-boxes, then we simply envelope each one of the clusters by the smallest box. This certainly implies that a bigger feasible set is considered with respect, for example, to consider convex hulls, but the choice is conservative, in the sense that less search space is pruned out.

Fig. 3.13 shows a set of feasible solutions and the corresponding boxes in a two-dimensional space. The figure highlights that the boxes cover some regions in which no feasible point was found. Additionally, it is worth noting that the boxes generated with this method can overlap.

REMARKS

The previous methods can be applied level by level, on the space $D_{L,i}$, or on the entire solution space up to the current level, D_i. In the former case, the boxing procedure is only applied to the level under consideration, on the variables $\mathbf{x}_{L,i}$, and

it generates the boxes in the space $D_{L,i}$. The union of those boxes is nonetheless the non-pruned space $\bar{D}_{L,i}$ defined in Eq. (3.5). To obtain the feasible set at level *i*, $\bar{D}_i$, the boxes shall be considered together with the boxes at the previous levels $1...i-1$. Since the boxes are defined on the spaces $D_{L,i}$, independently, there is no relation between one box at level *i* and another one at level *j*. The resulting feasible space, then, is generated by considering all the possible combinations of boxes for all the levels of the problem, or alternatively, considering a space which is generated by collecting all the boxes together (see Appendix C for details).

In the latter case, instead, the boxing procedure is applied directly to the whole space up to the level, D_i. This corresponds to what is presented in Eq. (3.9). Therefore, the resulting boxes are defined across all the levels, and each box has the same dimensionality of D_i. Therefore, the search space after the pruning $\bar{D}_i$ is simply the union of the hyper-rectangles defined by the boxes. This comes at the cost of having a clustering process on a space with higher dimensions.

3.5 Selection of the Planetary Sequence

A multiple swing-by trajectory is fully characterised given the sequence of planetary encounters, the departure date, the launch characteristics and a set of other parameters defining the timing and the characteristics of each swing-by.

The choice and the order of planets to swing-by are of great importance, as they change the trajectory completely. It follows that a proper selection of the planetary swing-bys is essential, and shall be done before or together with the optimisation of all the other continuous parameters characterising the trajectory. Furthermore, tackling the selection of a sequence with a standard optimisation approach is tricky, due to the discrete nature of the variables involved, and to the fact that, once a sequence is chosen, a global optimisation has to be performed in order to assess whether the sequence is promising or not. In addition, the number of different possible sequences of planetary swing-bys for an interplanetary transfer can be very high, which forbids to find an optimal trajectory for each one of them. In Chapter 5, the integrated search of the planetary sequence and the trajectory will be studied, by means of a novel approach based on Ant Colony Optimization. In this chapter, instead, the work is based on a two-level approach, in which the incremental pruning works at the lower level on a planetary sequence defined at the higher level.

Here we present a technique that provides a small set of candidate sequences, which can then be optimised through the incremental pruning. This technique is based on a simplified trajectory model, combined with a set of heuristic rules, which discard trivial sequences.

The simplified model allows a very quick assessment of all possible sequences. Since the simplified model is conservative, meaning that it gives an optimistic evaluation of each trajectory, it is possible to discard all the sequences which do not satisfy certain requirements (or are not feasible at all) in a very short time, without

the possibility of discarding promising ones. After this step, the complete model can be used to study the remaining sequences, through the incremental pruning, or any other global optimisation technique.

This method is applicable to any planetary or moon system, as long as orbital data of the bodies are known. For example, we can use the model for a MGA transfer in the solar system, where the main attractor is the Sun, and the swing-by bodies are the planets, or for a tour of the moons of Jupiter, where the latter is the main attractor and the moons are the swing-by bodies. In the following description of the algorithm, for the sake of simplicity, sequences of planets are sought, but the same method applies to sequences of moons, when a planet is the central body.

Two steps are used to assess the candidate planetary sequences: in the first one, the list of sequences is generated through some heuristic rules. In the second step, the feasibility of each sequence is assessed, so as to obtain the candidate sequences to be optimised.

3.5.1 Sequence List Generation

This step builds a list of possible swing-by sequences. Given departure and target bodies, the list is built incrementally by adding one planet at a time, considering constraints, which can be imposed on the planetary sequence itself. These constraints are based on heuristics, and allow discarding some trivial choices for the possible planetary sequences: examples are sequences which contain too many resonant swing-bys of the same planet, or contain swing-bys of farther planets (although energetically feasible, they can increase the transfer time too much).

The process of building the list of possible sequences is shown in Algorithm 3.3. The algorithm requires the departure planet P_0, as well as a list L_P containing all the planets that can be exploited for the swing-bys, and the destination planet P_f. The procedure makes use of three lists of sequences: $L_{inc,feas}$ contains the incomplete feasible sequences found so far (so sequences that respect the heuristic constraints but do not reach the target planet); L_{feas} contains the complete feasible sequences found so far; finally L_{temp} is a temporary list used within the loops. The list $L_{inc,feas}$ is initialised with only one incomplete sequence, which contains only the departure planet P_0; the other two lists are initialised to empty (lines 1 to 3).

During the algorithm, a temporary sequence $\mathbf{s}_{i,temp}$ is built by adding one planet at a time, chosen from the available planets in L_P, to each sequence in the list $L_{inc,feas}$ (line 7). The heuristic rules will tell whether to keep this new sequence or to discard it. The main loop in Algorithm 3.3 terminates when there are no more incomplete, feasible sequences. At the end, all the complete, feasible sequences are collected in list L_{feas}.

Algorithm 3.3. Sequence list generation.

1: $L_{inc,feas} = \{[P_0]\}$
2: $L_{temp} \leftarrow \varnothing$
3: $L_{feas} \leftarrow \varnothing$
4: **While** $L_{inc,feas} \neq \varnothing$, **Do**
5: **For** each sequence $\mathbf{s}_i \in L_{inc,feas}$, **Do**
6: **For** each body $P_j \in L_P$ for swing-by (and the arrival body), **Do**
7: $\mathbf{s}_{i,temp} \leftarrow [\mathbf{s}_i, P_j]$
8: Check the feasibility of $\mathbf{s}_{i,temp}$ according to the heuristics
9: **If** $\mathbf{s}_{i,temp}$ is feasible, **Then**
10: **If** $\mathbf{s}_{i,temp}$ is incomplete, **Then**
11: $L_{temp} \leftarrow L_{temp} \cup \{\mathbf{s}_{i,temp}\}$
12: **Else**
13: $L_{feas} \leftarrow L_{feas} \cup \{\mathbf{s}_{i,temp}\}$
14: **End If**
15: **End If**
16: **End For**
17: **End For**
18: $L_{inc,feas} \leftarrow L_{temp}$; $L_{temp} \leftarrow \varnothing$
19: **End Do**

The heuristic rules applied by Algorithm 3.3 are listed below. Some are based on the assumption that the orbits of the bodies are circular and coplanar, as will be explained in the following section.

- The number of swing-bys in a sequence is at most $n_{sb,max}$. This is needed to avoid the number of sequences to grow to infinity.
- The number of resonant swing-bys in a sequence is at most $n_{rsb,max}$.
- If the target body has a longer orbital radius than the departure's one, then the number of transfer legs going inwards is at most $n_{back,max}$. In the same way, if the target body's orbit has a smaller radius than the departure's one, then the number of transfer legs going outwards is limited.
- If the target body's orbit has a longer radius than the departure's one, then an inward transfer is possible only between two bodies whose orbits are spaced at most by $n_{backspacing,max}$ elements in the body list sorted by orbital radius. For example, in an Earth-Saturn transfer in the solar system, choosing $n_{backspacing,max} = 1$, an Earth-Mercury leg would make the sequence unfeasible, while an Earth-Venus leg is admitted. Equivalently, if the target body's orbit has a shorter radius than the departure's one, then an outward

transfer is possible only between two bodies whose orbits are consecutive in the list sorted by radius.

- If the target body's orbit has a longer radius than the departure's one, then no body with radius greater than the target one is allowed in the sequence. This is because if it is possible to reach an orbit whose radius is greater than the target, then it is possible to reach the target, too. Equivalently, if the target body's orbit has a shorter radius than the departure's one, then no body with a radius shorter than target's is allowed.

Depending on the problem, some or all of these heuristic rules can be used, and their parameters can be tuned accordingly.

After this first step, we have a number of possible sequences which can still be pretty high. Therefore, we assess the feasibility of each sequence with the following method.

3.5.2 Sequence Evaluation

This step assesses the feasibility (from an energetic point of view) of each sequence, and gives an approximated value of the relative velocity at the arrival planet. Since this quantity is often important when dealing with multi gravity assist transfers, it can also be used to rank each sequence in the list.

Each of the sequences found at the previous step is evaluated: since the number of sequences can still be pretty high, the idea here is to have a fast assessment of each sequence, using a reduced model, which can be considered an extension to what is proposed in [74]. In particular, the following assumptions were adopted:

- Orbits of all the bodies at which swing-by can be performed, and the spacecraft departure body, are considered circular;
- All the orbits and transfers are considered to lie in the same plane (planar system);
- No phasing is taken into account: a swing-by or rendezvous is possible every time the orbit of the spacecraft intercepts the orbit of the body;
- No other propelled manoeuvres are considered, other than the launch. Swing-bys are responsible for changing the heliocentric energy of the spacecraft; this makes this model unsuitable for evaluating the change in relative velocity in resonant swing-bys;
- No overturning of the outgoing heliocentric velocity vector after swing-by is allowed. This means that, if the rotation of the incoming relative velocity vector leads to an outgoing relative velocity vector which is on the other semi-plane with respect to the planet velocity, then the rotation which gives the maximum acceleration or deceleration of the spacecraft in the heliocentric reference is considered (see Fig. 3.14 and Fig. 3.15).

Fig. 3.14 and Fig. 3.15 show the triangles of velocity for a planar swing-by with a planet with circular orbit. The velocity of the planet $\mathbf{v}_P$ is purely transversal ($\hat{\boldsymbol{\theta}}$). The velocity vectors and angles referring to the incoming conditions at infinity before the swing-by are denoted by a superscript (–), while those after the swing-by are identified by (+).

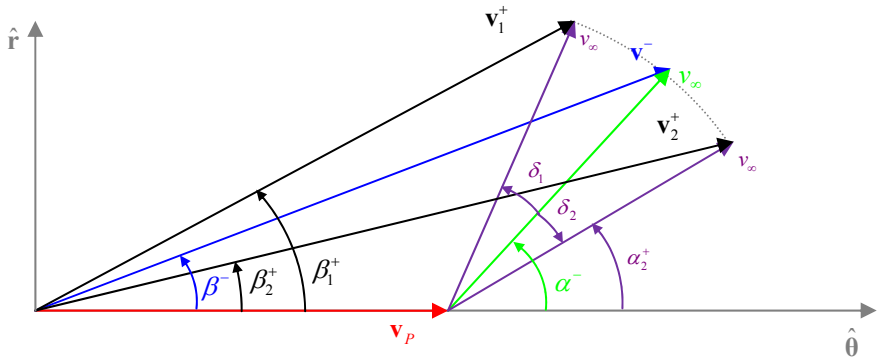


Fig. 3.14. Velocity triangles for a swing-by of a body with circular orbit (its orbital velocity is purely tangential).

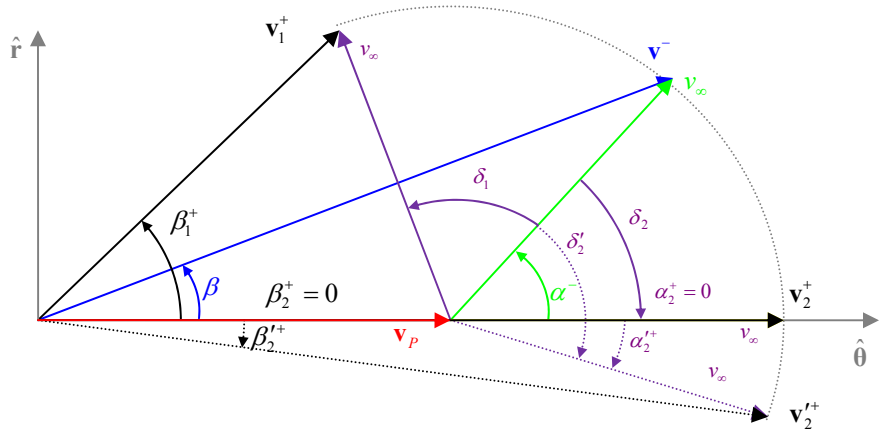


Fig. 3.15. Deflection correction in order to avoid overturning. Primed angles and dotted lines refer to the case in which overturning occurs.

The subscript (1) or (2) refer to the two possible outcomes of the swing-by, depending on the direction of deflection (see Section 2.2.2). In the case represented in Fig. 3.14, there is no possible overturning of the velocity vector: in fact, the deflection δ_2 originates the maximum increase in outgoing absolute velocity v_2^+. In the same way, the minimum outgoing velocity v_1^+ is obtained with deflection δ_2. In Fig. 3.15, instead, the maximum deviation δ_2' does not provide the maximum magnitude of the outgoing velocity vector. In fact, that is achieved by using a smaller deflection δ_2. This last one is the one used for computations within this model, to avoid overturning.

Given this procedure, and a sequence which is to be assessed, the trajectory model is used in the following way: depending on the problem, there can be a launch, or the initial conditions for the spacecraft are given at a certain planet. In the case of launch, the spacecraft is assumed to be at the departure body (thus on a circular orbit). The time of launch is not influent, as body's orbits are circular and

no phasing is considered. The launch excess velocity is tangential (to have the maximum or minimum variation of semi-major axis), and with the same verse of the velocity of the planet if the next planet in the sequence has a longer radius, otherwise is on the opposite verse. Instead, in case the initial velocity is given, then no other calculations or assumptions on launch are needed.

The resulting orbit (either after launch or obtained with the assigned initial velocity) is subsequently computed: if this orbit intercepts the orbit of the second body in the sequence, then a swing-by of that planet can be exploited. Assuming that the planet is in the correct position regardless of the arrival time, the incoming relative velocity is computed.

SWING-BY

Given the incoming conditions, a planar swing-by is fully determined by the pericentre and the direction of deflection. Petropoulos et al. [74] proposed to use the lowest allowed altitude for all the swing-bys, in order to maximise the heliocentric velocity variation (and thus maximising the change in energy). Still, fixing a particular value for the closest approach, there are two possible outgoing heliocentric velocities: one is higher (positive turning of velocity vector) and the other one is lower (negative turning) than the incoming velocity. The same authors made the following choice: if the orbit of the next different planet in the sequence has a longer semi-major axis, then the positive turning is chosen, otherwise the negative turning is considered.

This choice seems reasonable when assessing the feasibility of a given sequence, which is what is done in the work of Petropoulos. On the other hand, we think that this assumption is not suitable when an estimation of the final relative velocity is required. In fact, using the lowest possible altitude for the swing-by enables to determine whether it is possible to reach a given radius exploiting the swing-bys only, from an energetic point of view (i.e., neglecting plane change and phasing). Under these assumptions, though, the final relative velocity is just one of the many possible final relative velocities which can be achieved by changing the altitudes of the various swing-bys. If the problem requires to reach the final planet with the lowest possible relative velocity (a very common objective), or more seldom with a specific value of relative velocity, then this way does not provide any information on which sequence is possibly the best.

For this work, it was decided that several choices for the radius of the pericentre of the swing-by hyperbola have to be assessed, in order to have an estimation of the range of the final relative velocity which is achievable with the considered sequence. Certainly, it is not possible to consider a continuous set of values for the radius of pericentre, for each swing-by, as each combination of radii leads to a different trajectory.

Let $r_{p,i}$ be the i^{th} possible radius (expressed in radii of the planet). If the sequence under evaluation involves n_{sb} swing-bys, and the list of possible radii contains k elements, then the number of combinations of radii is $k^{n_{sb}}$. Considering that each swing-by with a fixed altitude has two possible outgoing velocities

(depending on the direction of the rotation of the relative velocity), then the total number of trajectories to assess for each sequence is $(2k)^{n_{sb}}$.

Despite the number of trajectories to evaluate can grow considerably with the number of swing-bys and number of possible radii, the evaluation of each trajectory is very fast, as this model does not imply the use of any computationally expensive algorithm (Lambert or Keplerian propagation), but the whole problem is solved analytically.

FEASIBILITY

Having fixed the combination of radii, the new orbit after the swing-by is computed and the algorithm continues looking for the intersection with the following body's orbit of the sequence. If the swing-bys allow the spacecraft to reach the last planet in the sequence, then we say that the sequence is *energetically feasible*, and the velocity relative to the last planet (v_∞) is computed and stored.

The procedure is repeated for each combination of radii/direction of deflection, and the minimum value of v_∞ (or the one closest to a target value) is stored. This value will be used as an indicator of the possible achievable value of v_∞.

3.5.3 Preliminary Test Case

A complete application of the sequence generation procedure to real test cases is in Chapter 4. Here we present a preliminary test case, where the method was tested on a MGA transfer from Earth to Saturn. Up to 2 resonant swing-bys were allowed in each sequence and 1 inward transfer was admitted. The launch excess velocity was set to 3 km/s. The resulting list of feasible sequences is given in Table 3.4.

Table 3.4. Feasible sequences for an Earth-Saturn transfer, according to energetic feasibility and heuristic rules.

1	E	V	E	E	M	S		
2	E	V	V	E	J	S		
3	E	V	E	E	E	M	S	
4	E	V	E	E	E	J	S	
5	E	V	E	E	M	M	S	
6	E	V	E	E	M	J	S	
7	E	V	E	E	J	J	S	
8	E	V	E	M	M	J	S	
9	E	E	V	E	E	M	S	
10	E	E	V	E	E	J	S	
11	E	V	E	E	E	M	J	S
12	E	V	E	E	M	M	J	S
13	E	V	E	E	M	J	J	S
14	E	V	E	M	M	M	J	S
15	E	V	E	M	M	J	J	S
16	E	E	V	E	E	M	J	S
17	E	E	V	E	M	M	J	S

The time needed to compute this list is about 1 second, using MATLAB® on a Pentium 3 Ghz computer. It is important to highline that, in the case that only the heuristic rules are applied, there are 249 possible sequences. The energetic feasibility criterion reduces this number to only 17, in a reasonably short time. It is also noticeable that sequence 2 in Table 3.4 is the sequence used by the ESA/NASA Cassini mission to Saturn [22]. This sequence is also the one that will automatically be found by ACO-MGA in Chapter 5.

3.6 Two Preliminary Case Studies

This section will present some preliminary results that were found using the incremental pruning. The sequence of planetary planets is pre-assigned.

We consider two basic problems with a single gravity assist manoeuvre: an Earth-Venus-Mars transfer and an Earth-Earth-Mars transfer. Despite the simplicity of these two test cases they are representative of two classes of MGA transfers and illustrate well the complexity of these kinds of problems. Moreover, the second problem will guide towards the definition of a particular partial objective function, of general validity for resonant swing-bys.

In these two tests, the search for the optimal solution at the second level is performed using boxing method 1 together with the affine transformation of the space, described in Appendix C, to which we refer for a complete description. The same appendix includes two additional test cases using the transformation (in combination with boxing method 3), including a transfer to Mercury, exploiting the sequence of swing-bys of the Earth and Venus twice. Although the use of the transformation led to some good results, it resulted to be unable to tackle problems with more complicated search spaces. Therefore, in this thesis, it will be limited to the tests in this section only. Boxing method 4, using the Mean Shift clustering algorithm, will be used in the case studies in Chapter 4.

In the following tests, the incremental approach was compared to the all-at-once approach with five different global optimisation methods, of which two are deterministic and three are stochastic. The two deterministic optimisers are DIRECT (Divided Rectangles, [114]) and MCS (Multilevel Coordinate Search, [115]). The stochastic optimisers are DEVEC, an implementation of Differential Evolution, an implementation of Particle Swarm Optimization, and the same Multi-Start method used for the search in the incremental approach. The optimisers were applied with different settings and with an increasing number of function evaluations. In the following, the optimisers are tested for 20000, 40000 and 80000 function evaluations.

The incremental approach uses at each level a search based on the Multi-Start technique (as described in Section 3.4.1). Since both the incremental algorithm and the global optimisation methods have a stochastic component, 20 runs were performed for each test case.

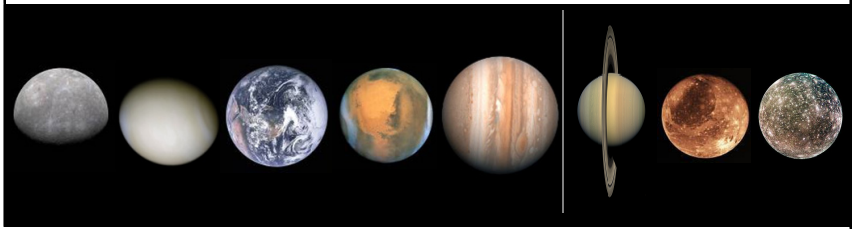
In all test cases the whole problem is decomposed into levels. After a set of feasible points is found and a set of boxes is generated, the affine transformation is

applied to the subspace of the level and the incremental approach proceeds by adding the next level.

We also present in Table 3.5 the data of the bodies – planets and moons – which will be used in this thesis. The planetary constant μ of the Sun is $1.3272 \cdot 10^{11}$ km^3/s^2.

Table 3.5. Physical constants of planets and moons used throughout this work.

Body	Gravity constant μ_P, km^3/s^2	Mean radius R_P, km	Orbital period P, d	Semimajor axis, km
Mercury (Me)	2.2033×10^4	2440	87.97	5.7909×10^7
Venus (V)	3.2486×10^5	6051.8	224.7	1.0821×10^8
Earth (E)	3.986×10^5	6378.2	365.25	1.496×10^8
Mars (M)	4.2828×10^5	3389.9	686.971	2.2794×10^8
Jupiter (J)	1.2669×10^8	6.9911×10^4	4331	7.7829×10^8
Saturn (S)	3.7931×10^7	5.8232×10^5	1.0759×10^4	1.4294×10^9
Ganymede (G)	9886.97	2634.1	7.15 (around Jupiter)	1.070×10^6 (around Jupiter)
Callisto (C)	7178.61	2410.3	16.69 (around Jupiter)	1.882×10^5 (around Jupiter)



3.6.1 Sequence EVM

The first test case consists of a transfer from the Earth to Mars exploiting a swing-by of Venus. For this test, no DSM along the Earth-Venus transfer leg is considered. Therefore the problem has dimensionality 6 and the bounds of the search space are reported in Table 3.6. Level 1 computes the first deep space flight phase, while the second adds the swing-by of Venus and the deep space flight to Mars. The objective function f is the total Δv, which is the sum of the relative velocity at departure and the DSM between Venus and Mars. The problem was initially analysed by running a Multi-Start optimisation on the whole domain. A local search was started from a total of 500 starting points, taken with Latin Hypercube, and the 10 best solutions are shown in Table 3.7. The total number of function evaluations needed to compute all the solutions was 494,233. The trajectory corresponding to the best solution is shown in Fig. 3.16.

DEVEC, Multi-Start optimisation and PSO were run on the whole problem for 200 consecutive times. In Table 3.8 it has been reported the percentage of times the stochastic optimiser finds a solution proximal to solution 1 in Table 3.7. In addition we report the percentage of times the stochastic optimisers find a solution that is better than the deterministic ones. The key point in the proposed incremental approach is not only to reduce the computational cost but also to increase robustness, i.e. increase the probability to find the global minimum.

The incremental approach starts at level 1 by looking for all local minima for the objective function f_1 which is the departure relative velocity v_0 at the Earth. A local search was started from a total of 20 random starting points and an equal number of boxes were generated. The size of the edges of the boxes was chosen as in Table 3.9.

Table 3.6. Bounds for the EVM test case.

Level	Variable	LB	UB
1	t_0, d, MJD2000	3650	9128.75 (3650 + 15 years)
	T_1, d	50	400
2	γ_1, rad	$-\pi$	π
	$r_{p,1}$, planet radii	1	5
	α_2	0	1
	T_2, d	50	700

Table 3.7. The 10 best solutions found with the all-at-once approach for the EVM problem.

Sol.	Δv, km/s	t_0, d, MJD2000	T_1, d	γ_1, rad	$r_{p,1}$, radii	α_2	T_2, d
1	2.9818	4472.013	172.29	2.9784	1	0.5094	697.61
2	2.983	4473.775	170.53	2.9859	1.0005	0.8611	698.15
3	2.9962	4475.217	171.12	2.853	1.076	0.7292	692.88
4	3.0393	4480.19	167.58	2.8044	1.1307	0.6371	692.57
5	3.1707	4482.079	174.65	-2.8195	1.1885	0.4608	629.93
6	3.1708	4482.145	174.60	-2.822	1.2033	0.4923	629.78
7	3.1719	4481.964	174.78	-2.8076	1.106	0.6224	630.77
8	3.1884	4471.355	171.44	-3.1416	1.019	0.5334	700.00
9	3.2217	3872.306	105.70	2.7087	1	0.545	628.02
10	3.2536	3872.891	104.68	2.6838	1	0.8006	627.22

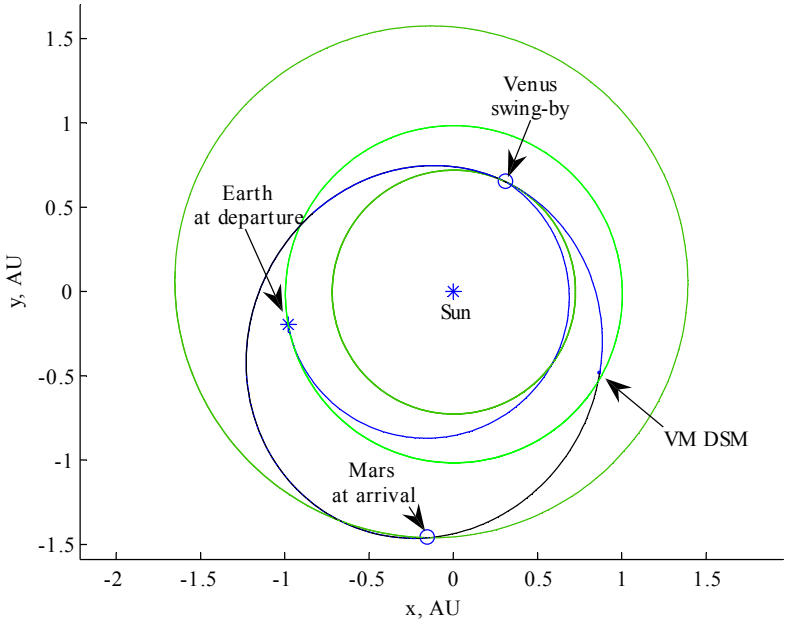


Fig. 3.16. Projection on the ecliptic plane of solution 1 in Table 3.7 for the EVM test case.

Table 3.8. Solutions and performances of different optimisers on the EVM transfer.

Solver	20000 evaluations	40000 evaluations	80000 evaluations
DIRECT, km/s	4.3760	4.3730	4.3730
MCS, km/s	6.7390	5.5240	5.4080
DEVEC, 200 runs			
< 3 km/s	6.5%	5.0%	7.0%
< DIRECT	99.5%	99.5%	99.5%
< MCS	100.0%	100.0%	100.0%
Multi-Start, 200 runs			
< 3 km/s	2.5%	3.0%	3.0%
< DIRECT	97.0%	99.0%	98.5%
< MCS	100.0%	100.0%	100.0%
PSO, 200 runs			
< 3 km/s	2.0%	2.5%	7.5%
< DIRECT	71.5%	73.0%	78.5%
< MCS	100.0%	96.0%	93.0%

Table 3.9. Box edges for the two variables of level 1.

Variable	Box edge
t_0, d	200
T_1, d	70

Fig. 3.17 shows the contour plot of the search space at level 1. The boxes which have been generated by the algorithm using method 1 are highlighted in blue, in semi-transparency. The size of the boxes is arbitrary and was set to a percentage of each dimension.

The Multi-Start search of the incremental algorithm was able to identify almost one local minimum for each synodic period. The number of evaluations to find all the 20 boxes was 516. After applying the affine transformation to level 1 and adding level 2, the whole reduced space was sampled with other 20 random starting points, and a local search was run from each one of them for a total of 8827 function evaluations. The result was that:

- 90% of the 20 best solutions found with the all-at-once approach have the values of level 1 variables included in one of the boxes;
- The best solution found with the incremental approach is the same as the best known solution, i.e. solution 1 in Table 3.7.

In addition, the incremental search has been run twenty consecutive times, obtaining always the same result and the same global minimum.

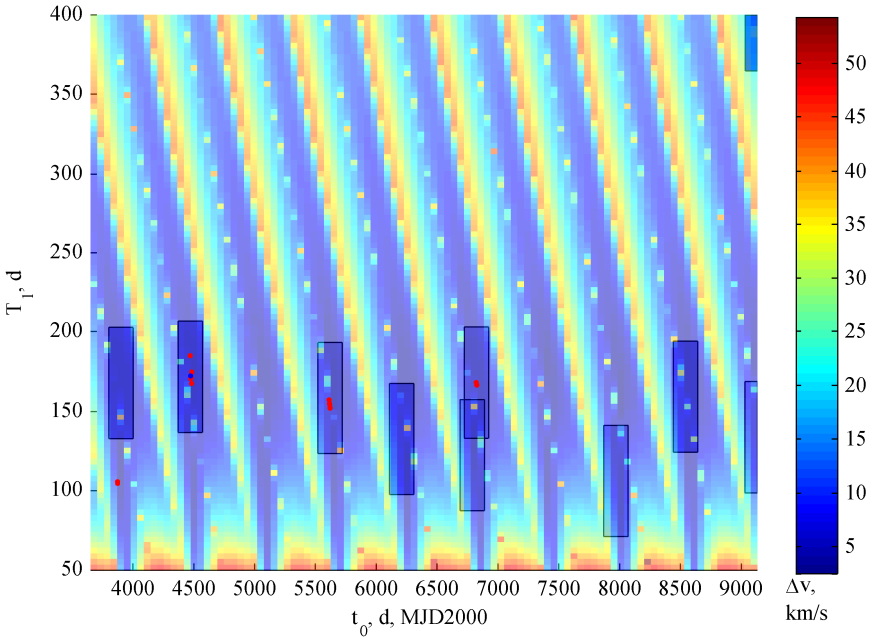


Fig. 3.17. Boxes found after analysing level 1. The space outside the boxes is pruned. The background gives an idea of the distribution of the local minima.

3.6.2 Sequence EEM

The second test case consists of an Earth to Mars transfer, through a swing-by of the Earth. This second case is significantly more complicated than the previous one due to the required optimisation of the Earth-to-Earth transfer in order to design a correct gravity assist manoeuvre. In fact, although this problem has two levels like the previous test case, the aim of this test is twofold: to demonstrate the effectiveness of the incremental approach, and to define a particular class of problem-dependent functions $\Phi_i(\mathbf{x}_i)$.

The Earth gravity assist is used to increase the kinetic energy of the spacecraft with respect to the Sun when the launch capabilities are limited. In order to gain the required Δv, the spacecraft has to reach the Earth with a relative velocity vector different from the one at departure. This is achieved with the DSM along the Earth-Earth transfer leg. Thus, the optimal design of the first leg is essential in order to exploit the encounter of the Earth properly, and gain the energy to reach Mars.

The launch velocity vector $\mathbf{v}_0$ depends on the launch capabilities, therefore its modulus was set at 2 km/s for this test case, while the declination δ and right ascension θ were left free.

Being v_0 constant, the solution vector has only 5 decision variables on level 1, and v_0 can also be removed from all the objective functions without loss of generality.

Table 3.10 presents the bounds for the variables of the problem. Once again the whole problem is decomposed into two sub-problems, corresponding to two levels. Level 1 consists of the Earth-Earth transfer, while level 2 computes the swing-by and the Earth-Mars transfer leg. The objective function f is the sum of the Δv of the two DSMs.

Due to the higher dimensionality of this test, for athe all-at-once approach 5000 starting points were used, leading to about $9 \cdot 10^6$ function evaluations. The 10 best solutions are shown in Table 3.11. Like in the previous case, the three stochastic optimisers and the two deterministic ones were tested for an increasing number of function evaluations and the results are reported in Table 3.12.

Table 3.10. Bounds for the EEM test case.

Level	Variable	LB	UB
1	t_0, d, MJD2000	3650	9128.75 (3650 + 15 years)
	δ, rad	$-\pi$	π
	θ, rad	$-\pi/2$	$\pi/2$
	α_1	0.01	0.99
	T_1, d	50	1000
2	γ_1, rad	$-\pi$	π
	$r_{p,1}$, planet radii	1	5
	α_2	0.01	0.99
	T_2, d	50	1000

Table 3.11. The 10 best solutions found with the all-at-once approach for the EEM problem.

Sol.	Δv, km/s	t_0, d, MJD2000	δ, rad	θ, rad	α_1	T_1, d
1	0.326	5430.17	-1.883	-0.0031	0.4691	500.065
2	0.333	6184.04	1.358	0.0266	0.5171	524.750
3	0.346	3650.00	1.341	-0.0032	0.4552	514.730
4	0.350	3770.54	1.789	0.025	0.2607	383.198
5	0.361	6318.71	1.793	-0.0642	0.2623	383.977
6	0.361	8732.38	-1.768	-0.0018	0.6846	888.721
7	0.368	6323.20	-1.446	0.0083	0.3073	385.862
8	0.374	6322.99	-1.446	0.0032	0.3163	387.371
9	0.376	5031.64	-1.774	0.0079	0.6778	886.791
10	0.378	5029.72	-1.774	0.0115	0.679	887.477

Table 3.11 continued.

Sol.	γ_1, rad	$r_{p,1}$, radii	α_2	T_2, d
1	-2.901	2.937	0.6195	307.900
2	3.142	3.022	0.4186	244.296
3	-2.982	4.438	0.4666	706.343
4	2.875	2.414	0.6642	711.414
5	2.749	2.833	0.0288	207.888
6	3.142	3.860	0.7118	766.862
7	-3.084	2.758	0.1549	242.328
8	3.119	2.815	0.1556	243.481
9	-2.800	4.173	0.3367	304.612
10	-2.762	4.046	0.228	302.372

Table 3.12. Solutions and performances of different optimisers on the EEM transfer, all-at-once approach.

Solver	20000 evaluations	40000 evaluations	80000 evaluations
DIRECT, km/s	2.7989	1.1870	1.1608
MCS, km/s	1.2070	1.2070	0.9944
DEVEC, 300 runs			
< 0.33 km/s	0.0%	2.7%	8.0%
< DIRECT	69.7%	87.7%	85.7%
< MCS	100.0%	86.3%	85.7%
Multi-start, 300 runs			
< 0.33 km/s	0.3%	0.0%	0.7%
< DIRECT	100.0%	98.3%	98.7%
< MCS	94.7%	98.3%	96.0%
PSO, 300 runs			
< 0.33 km/s	0.7%	0.3%	0.0%
< DIRECT	100.0%	91.3%	76.3%
< MCS	84.0%	91.3%	71.3%

The choice of the objective function f_1 for the incremental approach is trickier than in the previous case. In fact the cheapest way to perform an Earth-Earth transfer is to move from the Earth orbit as little as possible (or not move at all). Therefore, if the sum of the DSM and v_0 is chosen as objective to minimise, the optimiser returns solutions with no manoeuvre. These solutions, though, arrive at Earth with a relative velocity that is not suitable to exploit the swing-by properly. Furthermore, it is known from the physics of the problem that the zero-manoeuvre solution is a local minimiser even for the whole EEM transfer. Since the gravity assist manoeuvre requires an accurate timing to reach the swing-by planet with the right incoming conditions, its effect is to narrow down the basin of attraction of each minima. In fact, a gravity assist manoeuvre is more sensitive to a small variation of the variables than a direct transfer. Consequently, the gradient of the objective function in a neighbourhood of the local minima is higher and the basin of attraction is expected to be narrower. Now a zero-manoeuvre solution for the EEM

case physically corresponds simply to a delayed departure from Earth after the EE leg, with no gravity assist. All the zero-manoeuvre solutions, therefore, have a much wider basin of attraction. This can be easily verified by applying a general stochastic global optimiser to the whole EEM problem. The optimiser will return with a higher probability the zero-manoeuvre solutions if no special condition is imposed on the departure velocity at the Earth.

In order to minimise the Δv on the EM leg, the incoming velocity vector at the Earth should be such that the outgoing relative velocity vector is aligned with the velocity vector of the Earth (optimum increase in the kinetic energy).

A suitable criterion to optimise the first leg can be found by studying the characteristics of the relative velocity vector at the end of the Earth-Earth transfer. Fig. 3.18 represents the in-plane components (radial v_r and transversal v_θ) of the normalized incoming relative velocity vector for the best solutions found minimising the total EEM Δv with the all-at-once approach. On the same plot the objective function for the complete problem is also represented.

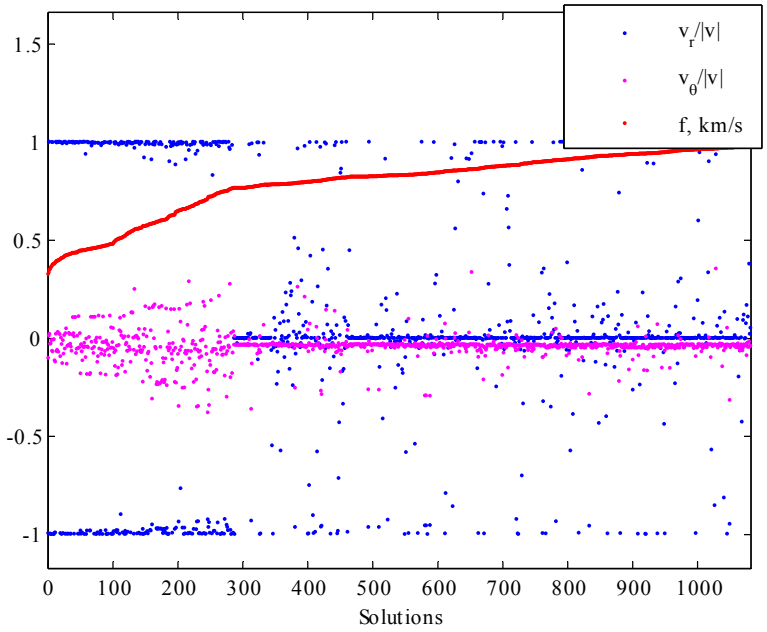


Fig. 3.18. Normalised in-plane components of the incoming relative velocity vector before the Earth swing-by, for the best solutions found, and corresponding objective value.

For the best solutions (from 1 to about 300), the direction of the relative velocity is almost completely radial, while for the rest of the solutions the radial component or the entire velocity drops to zero. Therefore, the following partial objective function can be chosen for all the levels in which there is the need for an outgoing velocity parallel to the velocity of the planet either to brake or to accelerate:

$$f_i = \beta \frac{v_{\theta,i}^{\ 2} + v_{h,i}^{\ 2}}{v_{r,i}^{\ 2}} + \sum_{l=1}^{i} \Delta v_l \,. \tag{3.17}$$

This function tries to minimise the DSM while maximising the radial component v_r of the relative velocity before the subsequent swing-by, with respect to the other components v_θ, v_h. The weight β was set to 1 km/s.

Although this criterion was derived for a specific case, it has general validity and applies to two classes of MGA transfers: aphelion rising gravity manoeuvres and perihelion lowering gravity manoeuvres.

The incremental approach was applied to level 1 starting a local search from 60 random points for a total of 13,547 function evaluations. Unlike in the previous case, the edges of the boxes on all the dimensions are a fraction of the search space, except for the edge along direction t_0, which spans the entire range (15 years). The reason is that the orbit of the Earth is almost circular: therefore a different position along its orbit has little influence on the arrival conditions at the end of the Earth-Earth leg. Thus, it is not possible to prune along t_0 at level 1, i.e. in the E-E leg. The size of the boxes was set according to Table 3.13.

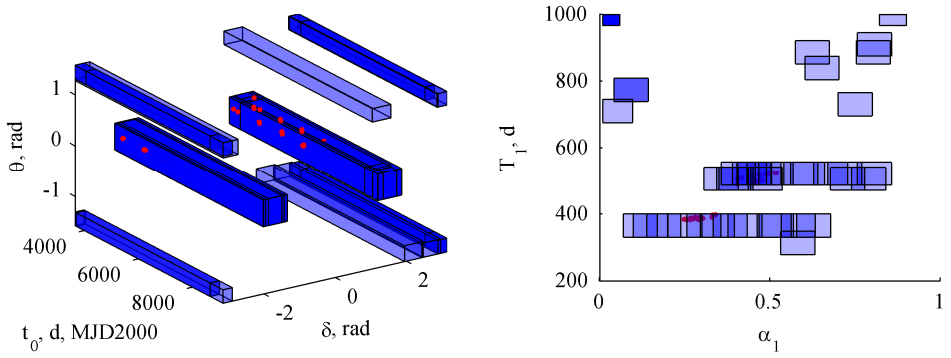
Fig. 3.19 (a) and (b) show the projection of the boxes along variables of level 1. The red dots represent the 50 best solutions found with the all-at-once approach. After applying the affine transformation to level 1 and adding level 2, the reduced search space was sampled with 30 points and a local optimisation was started from each one of them for a total of 32,544 function evaluations. The result was that:

- 86% of the 50 best solutions found with the all-at-once approach have the values of the level 1 variables included in one of the boxes;
- The best solution found with the incremental approach is the same best solution in the table.

Even in this case, the incremental approach was run for 20 times obtaining always very similar results. The globally optimal trajectory is represented in Fig. 3.20.

Table 3.13. Box edges for the two variables of level 1.

Variable	Box edge
t_0, d	Whole domain
δ, rad	$\pi/6$
θ, rad	$\pi/6$
α_1	0.1
T_1, d	70



(a) Projection along t_0, δ, θ (b) Projection along α_1, T_1

Fig. 3.19. Projection of the boxes in level 1 along the direction of the variables t_0, δ, θ (a) and α_1, T_1 (b). The stars are the 50 best all-at-once solutions.

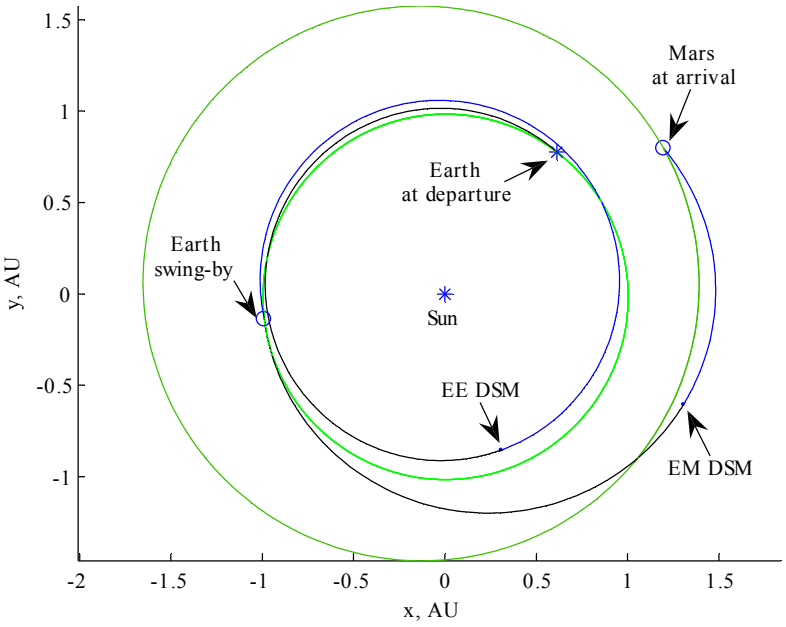


Fig. 3.20. Projection on the ecliptic plane of solution 1 of the EEM test case.

3.7 Comparative Results

In this section, some test cases will be presented, aimed at assessing the performances of the incremental pruning process over classic global optimisation techniques.

It will also be shown how the proposed incremental search performs an effective pruning of the search space, providing interesting results with a lower

computational cost compared to a non-incremental approach. In particular we compare the proposed incremental process, to the direct application of known stochastic and deterministic methods for global optimisation to the whole problem.

We will also show that the incremental pruning process can be coupled with several off-the-shelf search methods, and the performances of the different optimisers will be evaluated.

As resulted in the previous section, the affine transformation resulted to be efficient for relatively easy transfer problems. However, when more swing-bys are introduced and a wider range of parameters is chosen, the discontinuities in the affine space become significant, and therefore the use of the transformation seems not to improve the results. Furthermore, one of the key advantages of the pruning process is to isolate multiple families of solutions rather than a single global one. For these reasons, in the following test cases, the use of the transformation is substituted by independent optimisations on each feasible set.

3.7.1 Testing Procedure and Performance Indicators

When a new optimisation approach is proposed, it is good practice to test its performance against existing techniques on a known benchmark. In order for the tests to be significant, the testing procedure and the performance indicators need to be rigorously defined. The tests will compare the performance of a generic global optimiser, when applied to the search of the solution of a given problem by the all-at-once approach, against the performance of the same optimiser operating on the reduced search space after pruning. In fact, a key advantage of the proposed incremental pruning approach is to increase the probability to find sets of good solutions without increasing the computational cost. Furthermore, since the incremental approach makes use of stochastic-based techniques to identify the feasible set, some tests will demonstrate the reproducibility of the result of the incremental pruning itself.

A definition of a general testing procedure for global optimisation algorithms is presented in Appendix D, to which we refer. For the incremental approach, a number of performance indicators are defined that aim at establishing if the reduction of the search space achieved during the incremental search is reliable and efficient. It is here important to remind that the aim of the incremental approach is not to generate optimal solutions but to generate a set of sub-domains $\bar{D}_j \subseteq D$ bounding sets of locally optimal solutions. Therefore, the following indicators aim at measuring the ability of the incremental approach to repeatedly generate a tight enclosure of good solutions. Ideally, a good pruning would always yield few, small boxes enclosing the global optimum together with all the solutions satisfying $\delta_f < tol_f$. The performance indicators for the incremental pruning are presented in the following sub-sections.

Percentage of Inclusion of the Best Solutions

Through the all-at-once approach a number of solutions satisfying the condition $\delta_f < tol_f$ will be identified. Those solutions are considered to be neighbours of the

best one in the criteria space. The incremental approach is expected to identify, for every run, at least one box containing one or more of the solutions neighbouring the best one, i.e. one or more solution below the threshold. This indicator gives a measure of the ability of the incremental approach to identify good regions of the search space without discarding areas containing potentially good solutions.

PERCENTAGE OF PRUNED SPACE

This indicator measures the effectiveness of the reduction of the search space.

AVERAGE NUMBER OF BOXES

After the incremental pruning has completed the reduction of the search space, each block coming from the pruning can be further explored to find locally optimal solutions. A small number of boxes is therefore desirable, although multiple boxes can correspond to multiple equivalent launch opportunities. Thus, this indicator has to be used together with the percentage of inclusion of the best solutions. In the following we also considered the standard deviation on the number of generated boxes to provide an indication of the dispersion of the results.

COVERAGE

This indicator measures the ability of the incremental approach to perform a repeatable pruning of the search space. Given a box $B_{j,k}$ for run number k we compute the number of times that $B_{j,k}$ is covered partially or completely in all the other runs:

$$I_j(k) = \left| \left\{ i : \forall k, p, k \neq p \wedge B_{j,k} \cap B_{i,p} \neq \varnothing \right\} \right| \tag{3.18}$$

with $|\cdot|$ denoting the cardinality of the set. The other coverage index is the actual percentage of a block $B_{j,k}$ at run k that is covered by another block $B_{i,p}$ at run p:

$$\varsigma_j(k) = \frac{1}{n_{runs}} \sum_{p=1}^{n_{runs}} \sum_{i=1}^{q(p)} \mathrm{Vol}\left(B_{j,k} \cap B_{i,p}\right) \Big/ \mathrm{Vol}\left(B_{i,p}\right) \tag{3.19}$$

where $q(p)$ is the number of blocks resulting from run p, $\mathrm{Vol}(\cdot)$ is the volume of a given box.

3.7.2 Case Studies

The incremental algorithm was tested on the optimisation of two MGA trajectories: the first one is an Earth to Mars transfer, with a single gravity assist of the Earth (EEM sequence); the second is a transfer to Mercury, exploiting two swing-bys of Venus and one of Mercury (EVVMeMe sequence). These two test cases are representative of the class of MGA transfers with resonant swing-bys and well illustrate the complexity of this kind of problems. The incremental approach was compared to the direct solution of the whole problem (all-at-once approach) with three stochastic global optimisation methods, Multi-Start optimisation, Differential

Evolution and Monotonic Basin Hopping, and two deterministic global optimisers, DIRECT and Multilevel Coordinate Search.

For the incremental pruning, method 3 was used to generate boxes, and then no affine transformation was applied, which means that all the combinations of the boxes are to be searched.

SEQUENCE EEM

We use a similar test case to what was presented before, in Section 3.6.2, for the sequence EEM. The bounds of the problem are the same as in Table 3.10. The global objective function f is the sum of the Δv of the two DSMs, plus the difference between the spacecraft velocity and Mars velocity at arrival.

Table 3.14 reports the number of bins and the number of function evaluations for the pruning performed with MACS and MBH. MBH was run with a perturbation radius ρ_l equal to 10% of the range of the variable at level D_i. MACS was run with a population of 20 agents with $n_{popratio}$ of 10, and a tol_{conv} equal to 10^{-4} of the range of the variables at level D_i. For the restart of MBH we set the maximum number of trials to 30, while for MACS we partitioned only the second and fifth coordinate since they represent the most critical ones for the EE leg.

For the objective function f_1 of the incremental approach at level 1, the one in Eq. (3.17) was chosen, as it seemed promising according to previous test cases.

The threshold for level 1 was set to 1 km/s to leave some flexibility in the search for optimal resonant transfers. At level 2 the objective function is the total Δv, therefore it was not pruned.

A first test was run, applying two deterministic-based global optimisers, DIRECT and MCS, and three stochastic-based global optimisers, DE, MS and MBH to the entire problem to assess the performance of these global optimisers for an increasing number of function evaluations. The best known solution for this problem has a total Δv of 2.908 km/s (see Fig. 3.21), therefore, we set tol_f to 0.05 km/s.

Table 3.14. Number of intervals and function evaluations for the EEM case.

Level	Variable	Grid size	N. fun. eval. MACS	N. fun. eval. MBH
1	t_0, d, MJD2000	1	40000	40000
	$\bar{\theta}$	10		
	$\bar{\delta}$	5		
	α_1	5		
	T_1, d	30		
2	γ_1, rad	N/A	10000-40000	10000-40000
	$r_{p,1}$, planet radii	N/A		
	α_2	N/A		
	T_2, d	N/A		

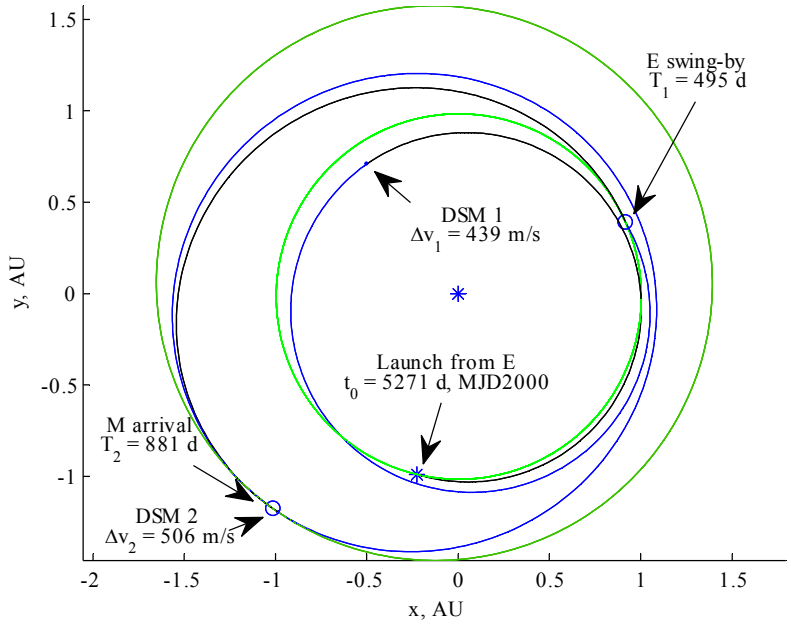


Fig. 3.21. Projection on the ecliptic plane of the best solution found by the incremental algorithm. The total Δv is 2.908 km/s.

Fig. 3.22 shows the distribution of the values of the variables at level 1 for 200 solutions with an objective function below 2.958 km/s. It is interesting to note that the solutions belong to different launch windows (different departure time) and have a departure velocity with respect to the Earth, which can be either against the velocity of the Earth ($\bar{\theta}$ close to 1), or perpendicular to it ($\bar{\theta}$ close to 0.5). It is therefore expected that both MACS and MBH will find distinct families of solutions when searching for the feasible set at level 1.

For the all-at-once test, we followed the procedure presented in Section 3.7.1, i.e. DE and MBH were run 100 times and the number of solutions with an objective function below or equal to the best-known solution was recorded. The DE algorithm was run with perturbation parameter $F = 0.8$, crossover parameter $C_r = 0.75$, search strategy *best*, and a population size of 90 individuals, while MBH was run with a ρ_l equal to 10% of the range of the variables. The results of the all-at-once test, summarised in Table 3.15, suggest that deterministic approaches, although their predictably yields the same solution at every run, do not provide satisfactory results. All stochastic approaches, on the other hand, are able to find, with almost 100% probability, better solutions than the deterministic ones. Therefore, although the probability of finding the best-known solution remains small for all stochastic methods, except MS, their use is advisable. Table 3.15, however, suggests that sophisticated global search methods, such as DE and MBH, are not the right choice; in particular, DE is the worst performing algorithm. The reason is the fast convergence of DE with the selected settings. As theoretically demonstrated in [56]

and [65], DE can converge to a fixed point in *D*, which is not necessary a local or global optimum. Once DE has converged, an increase in the number of function evaluations does not improve the performance. Furthermore, if the objective function is globally non-convex, i.e. presents multiple similar funnel structures, MBH may not be effective and DE could quickly converge but within a single funnel, mainly due to the selection heuristic. A simple Multi-Start algorithm, instead, can yield better performance provided that the local optimisation algorithm converges fast.

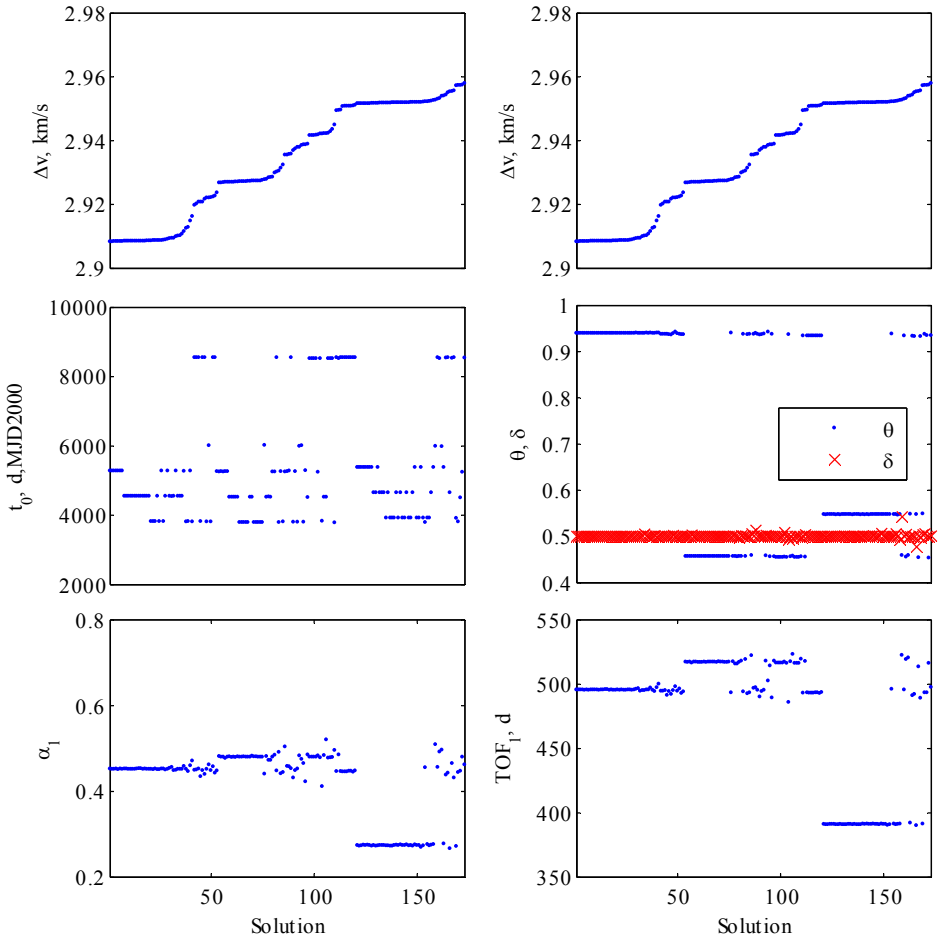


Fig. 3.22. Parameters of the first level for the local minima of the complete problem below 2.958 km/s. It is clearly visible that there exist solutions with objective value very close to the best known minimum, but having significantly different solution vectors.

Table 3.15. Comparison of different optimisation approaches applied to the EEM case all-at-once.

Solver	20,000 evaluations	40,000 evaluations	80,000 evaluations	160,000 evaluations
DIRECT, km/s	4.317	4.317	3.822	3.809
MCS, km/s	3.840	3.840	3.840	3.812
DE, 100 runs				
< 2.958 km/s	0%	7%	27%	27%
< DIRECT	68%	99%	100%	100%
< MCS	24%	85%	100%	100%
MBH, 100 runs				
< 2.958 km/s	1%	5%	18%	41%
< DIRECT	99%	100%	100%	100%
< MCS	96%	100%	100%	100%
MS, 100 runs				
< 2.958 km/s	22%	32%	52%	67%
< DIRECT	100%	100%	100%	100%
< MCS	100%	100%	100%	100%

Table 3.16. Incremental approach: performance on the EEM case over 100 runs.

Performance index	MACS	MBH
Inclusion of best solution	100%	100%
Pruned space (mean value)	90.43%	93.51%
Number of blocks, average	8.64	15.9
Number of blocks, standard deviation	1.64	0.082
Average coverage	88.58%	72.21%

The incremental algorithm applied to the first level yields the results in Table 3.16. The best solutions found with the all-at-once approach are always included in at least one of the boxes, which proves the reliability of the pruning, confirmed also by the value of the coverage indicators. At the same time, the percentage of pruned space is over 90% for both MACS and MBH. Therefore, it is expected that, when stochastic optimisers such as DE, MBH and MS operate on the reduced search space, the percentage of times they find solutions with a value lower than 2.958 km/s increases significantly compared to the results in Table 3.15. The performance of the incremental pruning based on MBH are not as good as the one of the incremental algorithm based on MACS, mainly because in the former case it generates about twice the number of boxes and with a lower average coverage.

Table 3.17 reports the performance of DE, MBH and MS on the reduced search space pruned at level 1. Differential Evolution was run with a reduced population of 45 individuals and with the same setting and strategy of the all-at-once case. If we look at the percentage of times the best solution is included in at least one box, the average number of boxes and the percentage of success for 10,000 evaluations on the box containing the best solution, we can conclude that the overall probability of identifying the best solution, after pruning, has considerably increased. On the other hand, the total number of function evaluations, accounting for both the incremental

pruning and the search in all the boxes, has decreased. Note that even if the number of function evaluations was the same, the total cost would be lower due to the lower cost of the evaluation at level 1. In fact, on an Intel Pentium 4 3 GHz running a MATLAB® coded algorithm under Microsoft Windows, the cost for a single function evaluation at level 1 is 2.33 ms while the cost at level 2 is 3.68 ms.

Table 3.17. Performance of DE and MBH on the box containing the reference solution over 100 runs.

Solver	10,000 evaluations	20,000 evaluations	40,000 evaluations
DE 100 runs			
< 2.958 km/s	52%	48%	56%
MBH 100 runs			
< 2.958 km/s	41%	52%	55%
MS 100 runs			
< 2.958 km/s	84%	97%	100%

Assuming 40000 function evaluations for the pruning at level 1 and 10000 function evaluations at level 2, for an average of 9 boxes, the total computational time for the incremental approach is 424,400 s against 478,400 s for the all-at-once approach with equal number of function evaluations. On the other hand, all the tested optimisers display a lower probability of success, in the all-at-once case, even for a higher number of function evaluations.

Fig. 3.23 shows a section of the domain D_1 along the first, second and fifth coordinate. The figure clearly shows the feasible set, identified by the clusters of solutions found by MACS, and the resulting boxes. The distribution of the feasible solutions is in agreement with what was found in Fig. 3.22. In particular the two clusters around $\bar{\theta} = 1$ and $\bar{\theta} = 0.5$, corresponding to the two optimal directions of launch, and the clusters around $T_1 = 500$ d and $T_1 = 300$ d. Also note that at level 1, as expected, all the departure dates are equivalent and cannot be distinguished.

SEQUENCE EVVMEME

For this test, the search was performed over the interval $t_0 \in [3457, 5457]$ d, MJD2000. In this interval of launch dates there exists a particularly good solution for the EVVMeMe that was considered as possible chemical option for the ESA BepiColombo mission [31]. This trajectory will be used as reference solution in the following. The upper and lower bounds for the other variables are reported in Table 3.18 and define a search space D with 14 dimensions.

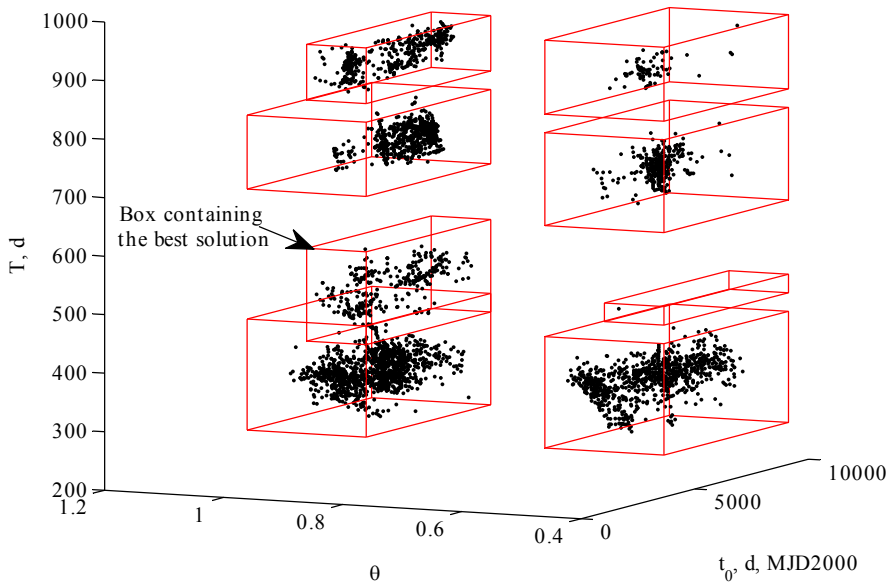


Fig. 3.23. **Section of the search space along the first, second and fifth coordinate. The figure shows the clusters of feasible solutions and the corresponding bounding boxes.**

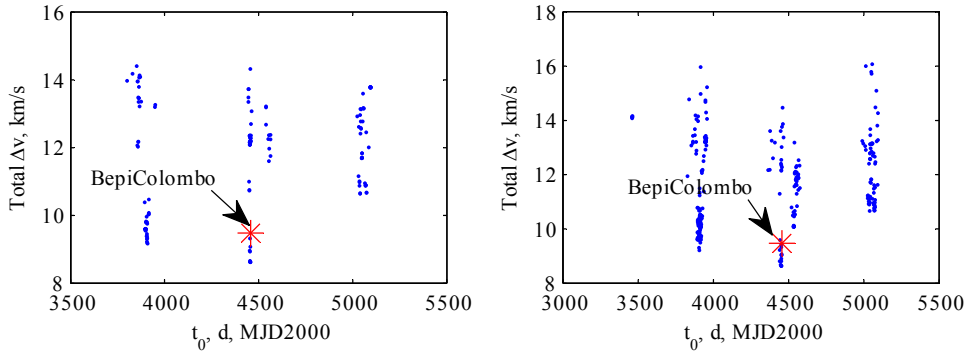
Table 3.18. Bounds for the EVVMeMe test case.

Level	Variable	Lower bound	Upper bound	N. bins	N. fun. eval. MACS	N. fun. eval. MBH
1	t_0, MJD2000	3457	5457	40	5000	15000
	T_1, d	90	180	10		
2	γ_1, rad	$-\pi/2$	$\pi/2$	3	70000	150000
	$r_{p,1}$, planet radii	1.01	2	1		
	α_2	0.01	0.6	4		
	T_2, d	448	673	5		
3	γ_2, rad	$-\pi/2$	$\pi/2$	3	140000	185000
	$r_{p,2}$, planet radii	1.01	2	1		
	α_3	0.01	0.9	4		
	T_3, d	90	220	5		
4	γ_3, rad	$-\pi/2$	$\pi/2$	N/D	10000-80000	10000-80000
	$r_{p,3}$, planet radii	1.01	1.1	N/D		
	α_4	0.01	0.5	N/D		
	T_4, d	260	352	N/D		

At first, we tested DE, MBH, DIRECT and MCS for an increasing number of function evaluations from 200,000 up to 1,600,000. DIRECT could not reach the highest number of function evaluations and was excluded from the comparison. DE and MBH were run 100 times, according to the testing procedure proposed above, and we recorded the number of solutions with an objective function below or equal to the reference solution. DE was run with $F = 0.8$, $C_r = 0.75$, *best* search strategy, and a population of 140 individuals, while MBH was run with a ρ_l equal to 10% of the range of the variables. The results are reported in Table 3.19. On such a complex search space a deterministic search like MCS cannot do better than 13 km/s. DE and MBH instead can find solutions that are better than 9.467 km/s, the ESA reference one. The probability of success, however, is limited to maximum 21% if DE and MBH are run on the whole search space. Furthermore, note that there is no statistical difference between the result of DE at 200,000, 400,000, 800,000 and 1,600,000 function evaluations, suggesting that DE converges too fast. On the other hand the aim of this preliminary set of runs is not to test DE but to have a standard of comparison for the application of the incremental pruning of the search space. All the solutions found with DE and MBH are represented in Fig. 3.24 (a) and (b) respectively, together with the reference solution, in the plane containing the departure date, in MJD2000, and the total Δv, in km/s. DE and MBH identify four main launch windows, with two of them containing solutions that are better than the reference one. Note that we compare solutions according to the total Δv only, while the reference solution was designed to fulfil also other requirements. Fig. 3.25 shows the reference solution (a) together with tree other solutions (b, c, d) with lower overall Δv all projected in the ecliptic plane.

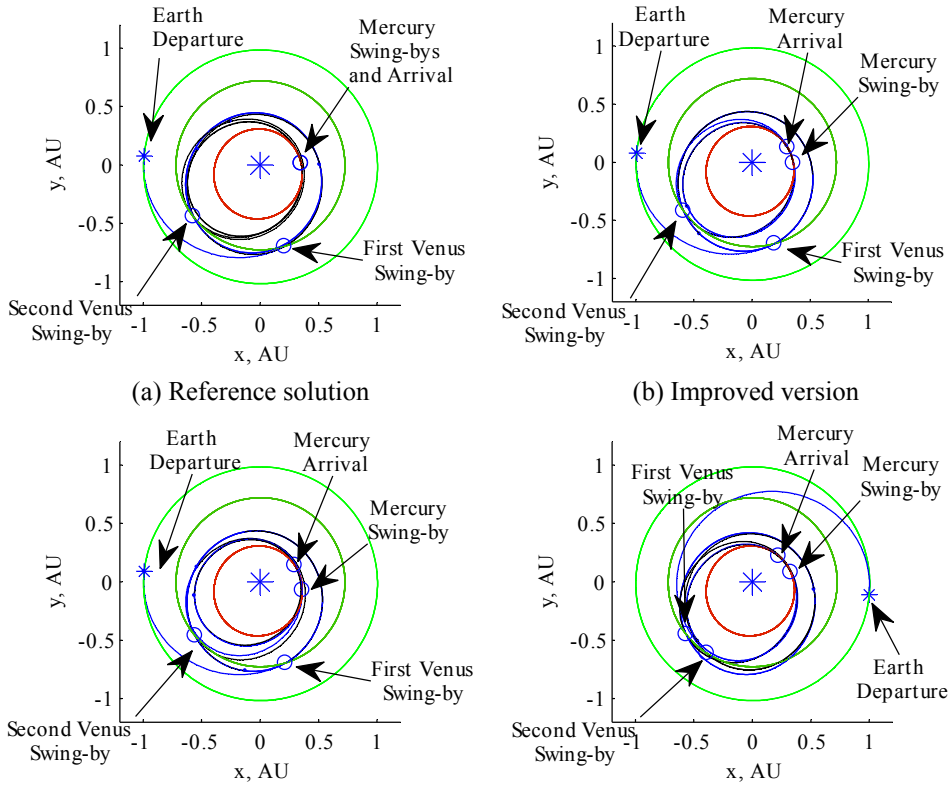
Table 3.19. Comparison of global optimisation methods applied to the EVVMeMe case.

Solver	200,000 evaluations	400,000 evaluations	800,000 evaluations	1,600,000 evaluations
MCS, km/s	14.35	13.05	13.05	12.01
DE, 100 runs				
< ESA (9.467 km/s)	16%	16%	21%	16%
< MCS	100%	81%	86%	61%
MBH, 100 runs				
< ESA (9.467 km/s)	4%	3%	7%	16%
< MCS	89%	78%	82%	100%



(a) DE (b) MBH

Fig. 3.24. Solutions found by DE (a) and MBH (b) over all 100 runs with all-at-once approach.



(a) Reference solution (b) Improved version

(c) Modified version (d) Different launch window

Fig. 3.25. Projection on the ecliptic plane of the reference solution and of three improved solutions found with the all-at-once approach. (a) The reference solution; (b) An improved version for the same launch window; (c) A modified version; (d) An improved solution for a different launch window.

The pruning was performed on all the variables but the swing-by variables $r_{p,i}$ because in the specified range it has a very low impact. The number of resonant orbits, i.e. fixed number of revolutions per leg, was pre-assigned. In particular we use the following resonance strategy: [0, 2, 0, 1, 2] where each number represents the number of full revolutions of the spacecraft around the Sun after the DSM and before meeting the next planet. For example, the EV leg is not performing any complete revolution around the Sun (after the DSM), while the VV leg performs 2 complete revolutions. The boundaries on the time of flight for each leg were computed as a function of the number of revolutions. In particular, assuming circular the orbits of the departure and destination planets of each leg, the time of flight is the period of the Hohmann transfer, between the two, times the number of revolutions while the upper bound is the period of the same Hohmann transfer times the number of full revolutions plus one. Note that, in principle, the trajectory model would not require the specification of the number of revolutions as the arc, propagated up to the DSM, can be as long as required. The resulting trajectory would have each DSM after the sequence of resonant orbits. This would make it not comparable to the ESA solution that has each DSM before the sequence of resonant orbits. However, in the velocity formulation model, the arc following the DSM is the solution of a Lambert's problem and the number revolutions has to be specified explicitly.

The number of bins and number of function evaluations for each level of the incremental pruning are shown in Table 3.18. The aim of the test is to show how a space reduction of all the levels from 1 to 3 can improve the search with DE and MBH at level 4. For this reason, level 4 was not pruned. In general we can consider that the last level is the least significant for the pruning. MBH was run with a perturbation radius ρ_l equal to 10% of the range of the variable at level D_i. MACS was run with a population of 20 agents with $n_{popratio} = 10$, and a tol_{conv} equal to 10^{-4} of the range of the variables at level D_i. For the restart of MBH we set the maximum number of trials to 30, while for MACS we use no partitioning at level 1 and we partitioned only γ and the time of flight at each subsequent level.

Special partial pruning criteria were used for the legs ending with Venus and Mercury. In particular, for the arrival conditions at Venus we use objective function in Eq. (3.17), with $\beta = 1$ km/s, while for the arrival conditions at Mercury (last level) the objective function was:

$$f_3 = v_\infty + \sum_{k=1}^{3} \Delta v_k \tag{3.20}$$

where v_∞ is the spacecraft velocity relative to Mercury at arrival. The pruning functions were selected based on the required effect of the gravity assist manoeuvres. In particular, Venus gravity manoeuvres are expected to maximise the change in the perihelion while the manoeuvres at Mercury, combined with the DSM, are supposed to minimise the relative velocity at Mercury. The pruning thresholds are on the value of the partial objective function, and are 4, 4.5 and 7

km/s for level 1, 2 and 3 respectively. The first threshold is set according to launch capabilities, the second accepts a DSM no larger than 0.5 km/s for the VV resonant flyby, while the third threshold estimates the arrival v_∞ at Mercury to be lower than 6 km/s. Table 3.20 reports the performance of the incremental pruning on the EVVMeMe case with pruning up to level 3 included. Note that the reference solution and the best solutions (all the ones better than the reference) are included in at least one of the boxes in the majority of the cases, in particular when the MACS is used to look for the feasible set. The higher reliability of MACS compared to the modified MBH is marginal and not statistically significant. On the other hand, MACS tends to generate a higher number of boxes. Note that the significance of the variance in the table has to be taken with care because the process is not necessarily Gaussian. The only thing that can be said is that already at level 3 the number of boxes is high despite the fact that many of them were clustered. The last row of Table 3.20 reports the coverage metric. The reproducibility of the pruning is quite good for both algorithms. Table 3.21 shows the percentage of success of DE and MBH on the residual space, in particular on the most promising box. As can be seen the increase in performance is considerable, reaching almost 100%, fully justifying the space reduction obtained with the incremental approach. Finally, Table 3.22 reports the time of evaluation of the partial objective function at each level. Again, the lower cost of the partial objectives at lower levels leads to an overall gain in computational speed during the incremental pruning since a good deal of the function evaluations are used when exploring the lower levels. In particular, in this case 215,000 function evaluations were used to prune levels from 1 to 3 with MACS and 350,000 with MBH but the cost of each function evaluation was between 1/3 and 1/10 of the cost at level 4.

Table 3.20. Incremental approach: performance on the EVVMeMe case over 100 runs.

Performance index	MACS	MBH
Inclusion of best solution	100%	94%
Inclusion of ESA solution	91%	89%
Pruned space (mean value)	97.81%	97.8%
Number of blocks, average	53.1	45.21
Number of blocks, standard deviation	19.48	15.21
Average Coverage	80.1%	81.8%

Table 3.21. Performance of DE and MBH on the box containing the ESA solution over 100 runs.

Solver	10,000 evaluations	20,000 evaluations	40,000 evaluations	80,000 evaluations
DE 100 runs				
< ESA (9.467 km/s)	89%	91%	99%	99%
MBH 100 runs				
< ESA (9.467 km/s)	32%	48%	72%	88%

Table 3.22. Average time to evaluate the partial objective functions, for each level.

Level	1	2	3	4
Time, ms	1.69	5.2	5.7	6.94

3.8 Discussion

In this chapter, we presented a simple approach to the design of multiple gravity assist trajectories. The approach decomposes the whole trajectory in sub-problems of lower dimension and complexity and proceeds to reduce the search space incrementally, adding one leg at a time. This incremental space reduction, tested on a number of cases, demonstrated to significantly increase the chance to find good solutions at a relatively low computational cost: the probability of success of all tested optimisers on the pruned space is from two to six times higher and with a similar overall number of function evaluations. Furthermore, due to reduced computational cost of the evaluation of the partial objective functions, the total computational time of the incremental approach is lower even for the same number of function evaluations. Therefore, we obtained an increase in the reliability of the optimisation process (i.e. higher success rate) with a reduction of its computational cost. An additional advantage of the proposed incremental approach is the generation of a set of feasible solutions rather than a single optimal one. The feasible set corresponds to families of possible mission opportunities for different launch dates. Thus, the decision maker, or mission designer, is presented with multiple options together with an enclosure of their neighboring solution space. The amount of available information is, therefore, higher and would allow an easier identification of baseline and backup designs together with their robustness. In fact, the size of the neighborhood, or subset of solutions below a given threshold, can be seen as a measure of their sensitivity to small changes in the design parameters. These features will be extensively explioted for the case studies in the following Chapter 4.

The critical aspects for an efficient implementation of the incremental solution of MGA trajectories are: the definition of specific partial objective functions for each incremental level and the definition of appropriate pruning criteria. Furthermore, the search for the feasible set can substantially change the performance. The two methods presented in this chapter, MACS and MBH, performed substantially in the same way for all the test cases. The main difference is that, while MACS can be used with discontinuous and noisy functions, MBH needs a smooth and continuous function. In both cases the incremental algorithm remains unchanged, therefore it is recommended to start with a simple search procedure, even a simple Multi-Start algorithm, if the partial objective functions are smooth and differentiable.

Although the incremental procedure is generally applicable to all the trajectory models presented in this chapter, it was tested on the one model built with the velocity formulation because this model leads to an exponential growth of the possible alternative path with the number of swing-bys. Furthermore, we presented

a forward incremental procedure but, following the same principle, the procedure can be performed backward or forward and backward at the same time.

4

APPLICATION TO REAL CASE STUDIES

The incremental pruning is used here to design parts of the ESA missions Laplace and BepiColombo. It will be shown that the method is suitable to find solutions that not only minimise the cost, but also meet other, more complex requirements. Moreover, the feasible sets and local optima found will be used for further investigations on the problem, therefore showing the value of the tool for interplanetary mission study and design.

4.1 Introduction

The test cases in this chapter aim at showing the applicability of the proposed incremental approach to the design of real missions. It will be shown how the incremental pruning method can be extended, without substantial modifications, first of all to take into account additional cost functions such as the relative velocity at the target planet, v_∞, and the total time of transfer, then to target planets as well as specific orbits. Another important point in preliminary mission design is the need for a set of trajectories that can be used for a trade-off analysis. In other words, it is required not only to find the globally optimal solution, but also a number of low-laying feasible local optima. In this chapter it will be shown that the incremental pruning responds to this requirement.

In contrast to the test cases in the previous chapter, here the aim is not to provide a comparative performance assessment of the incremental approach; instead, the computed solutions will be compared to the existing baseline solutions for each mission. An analysis of the solutions will follow to highlight their main features, as well as to draw some general conclusions that could help the search for similar solutions in similar applications. This will show that the incremental pruning is also useful for studying the MGA transfer problem.

The test cases presented in this chapter are part of two missions currently under design: Laplace [35] and BepiColombo [31]. The initial conditions and objectives of the problems proposed were provided by the Mission Analysis section at ESA-ESOC, and reflect the actual design state of the mission at the time when these tests were performed[1]. Therefore, they are very well representative of the real trajectory design problem.

The tests were performed using 2 different machines:

- Intel Core 2 Duo T5500 1.66 GHz equipped with 2 Gb of RAM, and running Windows Vista;
- Sun Ultra 45 Workstation (UltraSPARC IIIi 1.6 GHz), and running Sun Solaris.

Implementation details are given in Appendix A.

4.2 Laplace Mission

Laplace is a joint ESA-NASA-JAXA interplanetary mission to Europa and the Jovian system. The high-level scientific goals of the mission are to determine the conditions for the formation of the Jupiter system, understanding how Jupiter works and whether Europa is habitable [116]. The launch time-frame is 2016-2018. A NASA spacecraft and an ESA spacecraft are planned to be launched separately: the former will perform science around the rocky moons Io and Europa. The ESA orbiter, instead, will target the icy moons, Ganymede (G) and Callisto (C). Secondary objectives for this spacecraft will be a fly-by of Europa (releasing an Europa penetrator), a fly-by of a small body before Jupiter arrival, and the release of the JAXA magnetospheric orbiter [117].

A baseline scenario was proposed with a multi-gravity assist transfer from the Earth to Jupiter, with sequence Earth-Venus-Earth-Earth-Jupiter and an arrival date in October 2024. The baseline scenario foresees that, at arrival, a first Ganymede gravity assist (designated as GGA1 in the remainder of this chapter) will be performed to insert the spacecraft into a high elliptical orbit around Jupiter. A perijove raising manoeuvre will be required, to avoid the radiations at short distance from Jupiter. Three resonant swing-bys of Ganymede will follow (GGA2-GGA5) to reduce the orbital period, reduce the inclination, and reduce the infinite velocity with respect to Ganymede. At this point, a multiple swing-by tour involving Ganymede and Callisto will be exploited to reach Callisto with a low relative velocity. The spacecraft will perform a number of swing-bys of the moon to achieve low altitude coverage of the majority of the surface of Callisto, without orbiting around it.

Finally, a transfer to reach Ganymede with low relative velocity will precede the Ganymede orbit insertion (GOI), during which science on Ganymede is

[1] The test cases and studies that will be presented in this chapter were found during a visit of the author at the Mission Analysis Office (OPS-GFA) of ESA-ESOC (European Space Agency, European Space Operations Centre) in Darmstadt, Germany.

performed. Firstly, the magnetosphere is analysed while on an elliptic orbit, then the orbit is circularised into a low altitude orbit.

The analyses in this chapter will focus on two particular segments of the ESA orbiter mission: the multiple resonant swing-bys of Ganymede (GGA2-GGA5) and the transfer from Ganymede to Callisto.

4.2.1 Resonant Swing-bys of Ganymede

This section will describe the optimisation of the resonant swing-bys of Ganymede (GGA2-GGA5) to reduce the velocity relative to Ganymede (v_∞). The initial conditions of the tour at Ganymede are given by incoming conditions at Jupiter at the end of the interplanetary transfer from the Earth.

The initial conditions are summarised in Table 4.1. The spacecraft is assumed to be at Ganymede (zero sphere of influence) and the position is known through the ephemerides of Ganymede. The orbit of the spacecraft before the first swing-by, and the orbit of Ganymede are represented in Fig. 4.1. The reference frame is the inertial Jupiter equatorial reference frame at epoch.

Table 4.1. Initial epoch, velocity relative to Ganymede, orbital inclination and period for the GGA2-GGA5 transfer case.

Variable	Value
t_0	9230.8850 d, MJD2000 (10 April 2025)
$\mathbf{v}_{\infty,r\theta h}$,[2] km/s	[-4.9487, 3.1372, -1.2934]
i, deg	5.29
P, d	179

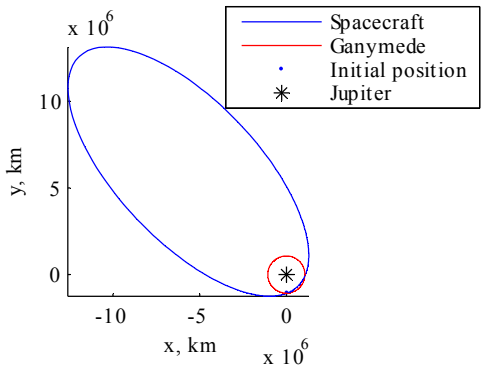


Fig. 4.1. The initial orbit of the spacecraft (before the first G swing-by), in blue, and the orbit of Ganymede (in red).

[2] The reference frame *r*-*θ*-*h*, sometimes referred to as *RST*, is the one having radial, transversal and out-of-plane unit vectors as axes.

The initial relative velocity with respect to Ganymede is 6 km/s in magnitude. The purpose of the resonant Ganymede sequence is to reduce the v_∞ to 5 km/s, the inclination to 0 deg, and the orbital period to 3 periods of Ganymede ($P_G = 7.15$ d), i.e. 21.45 days, at minimum Δv and time cost.

INCREMENTAL SOLUTION

The application of the incremental approach for resonant swing-bys is tricky. In an MGA transfer without any resonant swing-by, the DSMs are mostly used to change the v_∞ with respect to a given body, but very often ballistic transfers are still possible (with some exceptions: see last sub-section of Section 4.2.2). This means that minimising the Δv and pruning on Δv is often a good strategy to find ballistic or quasi-ballistic solutions. Swing-bys can also change the v_∞ with respect to other planets. DSMs can then be introduced to correct slightly the v_∞, without changing much the shape of the trajectory. This is also the reason for which the v_∞ may not be taken into account before the last level of the transfer (the validity of this will be shown in Section 4.2.2, though a non-resonant MGA transfer case).

In resonant swing-by sequences, each swing-by can change the orbital parameters of the spacecraft (e.g. inclination, semi-major axis and eccentricity) but not the v_∞ with respect to the planet. Deep space manoeuvres are used to approach the planet with the appropriate v_∞; therefore, the Δv magnitude of each DSM is not enough to prune the search space incrementally, leg by leg. If only the Δv's were considered, the pruning would not be effective, consequently ending up with a large number of clusters covering great part of the search space. A better choice is to introduce some the target parameters of the problem (e.g. inclination and v_∞) into the objective and the pruning functions together with the Δv's. The objective function can be defined as the sum of three terms:

$$f_l = g_{v_\infty,l}(v_\infty) + g_{i,l}(i) + g_{\Delta v,l}(\Delta v) \tag{4.1}$$

where $g_{v_\infty,l}$ is a function of the v_∞ at the end of level l, $g_{i,l}$ depends on the inclination of the orbit at the end of the trajectory and $g_{\Delta v,l}$ is a function of the total Δv of the trajectory up to the level l. The functions g have the general form:

$$\begin{aligned} g(x) = &\left(\frac{\frac{100}{s} \cdot \arctan\left(\left((x - u_u) - \frac{s}{20} \right) \right)}{\pi} + 0.5 \right) \cdot \exp\left(\frac{x - u_u}{sr} \right) + \\ &+ \left(\frac{\frac{100}{s} \cdot \arctan\left(-\left((x - u_l) + \frac{s}{20} \right) \right)}{\pi} + 0.5 \right) \cdot \exp\left(-\frac{x - u_l}{sr} \right) \end{aligned} \tag{4.2}$$

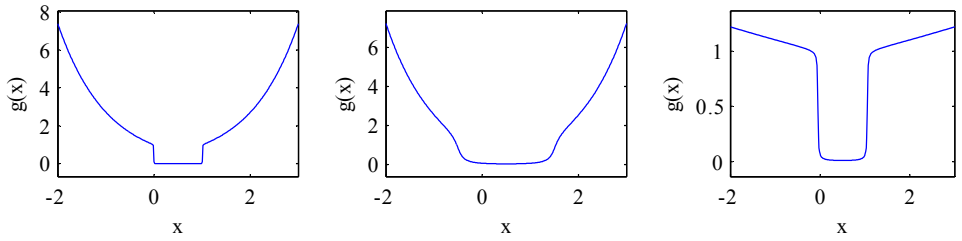
and depend on the shaping parameters u_l and u_u, r and s. As shown in Fig. 4.2, the general shape is a composition of a low laying basin within the interval $[u_l, u_u]$, with an exponential growth outside this interval. The basin has a depth of 1, and the bottom of it is almost flat and very close to zero.

As it can be seen from Fig. 4.2, s is changing the shape of the edges of the function at the borders of the basin, while r is a scaling factor for the exponential part. If either u_l or u_u are set, respectively, to plus or minus infinity, one obtains the functions in Fig. 4.3, where the basin is unbounded on either side.

The resonant sequence is made of 3 swing-bys of Ganymede, giving an incremental problem with 3 levels. Since the initial conditions are given at Ganymede, before its swing-by, then the solution vector is the one in Eq. (2.19) in Section 2.2.4:

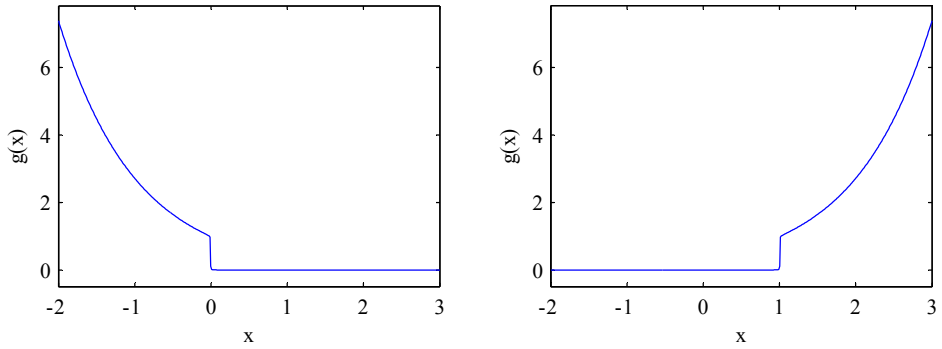
$$\mathbf{x} = \left[\gamma_1, r_{p,1}, \alpha_1, T_1, \gamma_2, r_{p,2}, T_2, \alpha_2, \gamma_3, r_3, \alpha_3, T_3\right]$$

Bounds for this problem are in Table 4.2, where $i = 1,2,3$. The values of the parameters u_l and u_u in the objective function, for each level, are in Table 4.3. Parameters r and s were kept constant for all the levels, as shown in Table 4.4.



(a) $r = 10$, $s = 0.1$ (b) $r = 0.1$, $s = 10$ (c) $r = 10$, $s = 1$

Fig. 4.2. Shape of the function $g(x)$ (Eq. (4.2)) for $u_l = 0$, $u_u = 1$, and different values of r and s.



(a) $u_l = -\infty$ (b) $u_u = +\infty$

Fig. 4.3. Shape of the function $g(x)$ (Eq. (4.2)) when $u_l = -\infty$ (a) or $u_u = +\infty$ (b).

Table 4.2. Bounds for the GGGG transfer case.

Variable	LB	UB
T_1, d	40	100
T_2, d	20	60
T_3, d	20	21.5
$r_{p,i}$, R_P	1.1	3
α_i	0.01	0.9
γ_i, rad	$-\pi/2$	$\pi/6$

Table 4.3. Parameters *a* and *b* for the objective function.

Level l	$g_{v_\infty,l}(v_\infty)$		$g_{i,l}(i)$		$g_{\Delta v,l}(\Delta v)$	
	u_l, km/s	u_u, km/s	u_l, deg	u_u b, deg	u_l, km/s	u_u, km/s
1	–∞	5.8	–∞	5.5	–∞	0.07
2	–∞	5.6	–∞	5	–∞	0.14
3	4.98	5.02	–0.02	0.02	–∞	0.16

Table 4.4. Parameters *r* and *s* for the objective function.

l = 1, 2, 3	r	s
$g_{v_\infty,l}(v_\infty)$	10	0.5
$g_{i,l}(i)$	1	0.1
$g_{\Delta v,l}(\Delta v)$	10	0.1

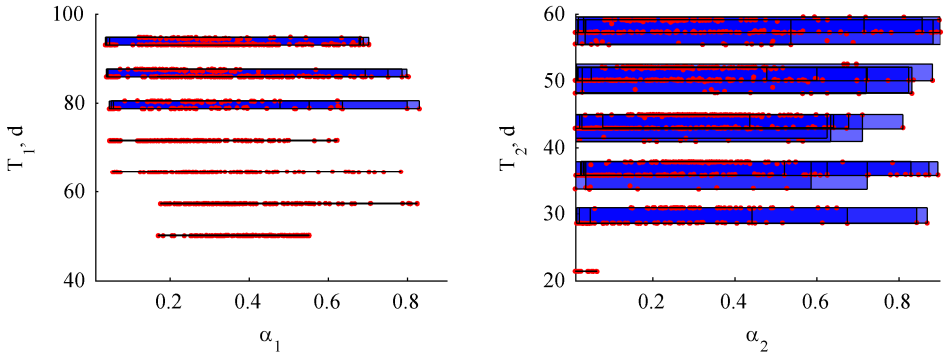
For the incremental pruning process, the Multi-Start optimisation was used, because it was shown to be reliable and simple to use in terms of tuning parameters. The solution is considered feasible (and thus the local optimisation stopped) as soon as the objective value falls below 0.5. This ensures the solution to be inside the low-laying basins of all the three functions $g_{v_\infty,l}(v_\infty)$, $g_{i,l}(i)$ and $g_{\Delta v,l}(\Delta v)$. For both level 1 and 2, the pruning function was set as:

$$\Phi_i(\mathbf{x}_i) = f_i \le 0.5 .$$

and the boxes are generated by the clustering technique using the Mean Shift algorithm. This technique was chosen, over the others, because, although produces boxes that include some parts of the non-feasible space, it is suitable to identify multiple solution families (as it will be shown throughout the chapter).

RESULTS

Fig. 4.4 (a) and (b) show the projections of the solutions and the clusters along $[\alpha_1, T_1]$ (a) and $[\alpha_2, T_2]$ (b) after pruning level 2. It appears that solutions are clustered by time of flight. This is due to their different resonances, as it will be explained shortly.



(a) Projection along α_1, T_1 (b) Projection along α_2, T_2

Fig. 4.4. Projections of solution set and feasible regions after pruning level 2.

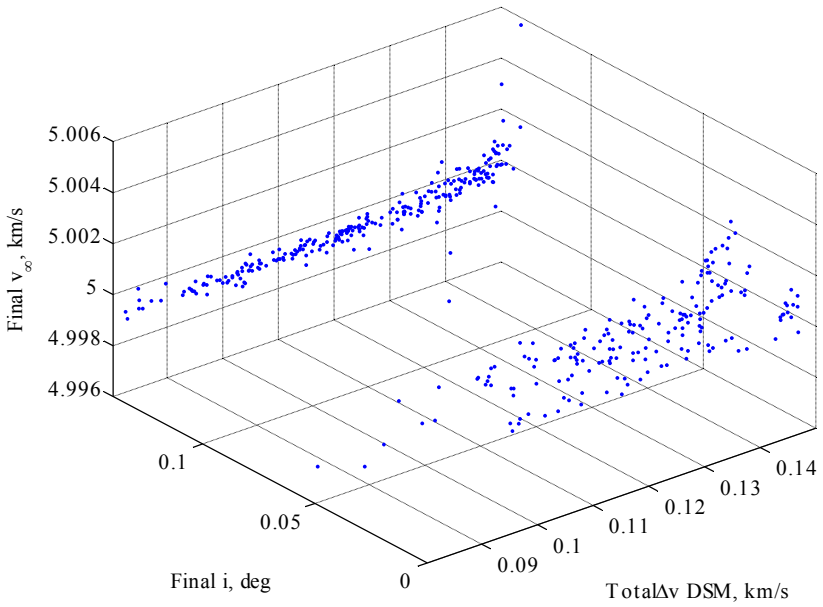


Fig. 4.5. Total Δ*v*, inclination and final relative velocity at G for the GGGG sequence. Only solutions with Δ*v* lower than 150 m/s are shown.

Solutions found after optimising level 3 are shown in Fig. 4.5, in terms of Δv, v_∞ and inclination i at the last encounter. The figure presents only solutions having $\Delta v < 150$ m/s.

Note that all the solutions have a v_∞ which is very close to the objective of 5 km/s. This means that this requirement is overall fulfilled with good accuracy. Note also that there are two solution families. One family converged to a value of i very close to 0 deg, while a second family converged to about 0.12 deg.

Fig. 4.6 shows the Δv and total time of flight for the solutions with $i < 0.15$ deg and $\Delta v < 150$ m/s. As a comparison, the red dot represents the baseline solution designed by the European Space Agency. The plot shows that alternative solutions exist with a lower Δv cost and comparable transfer time together with even cheaper solutions for longer time of flight. A green line represents the envelope of the solutions that are Pareto-optimal. Note that, because of the resonance constraint, the total time of flight can assume only a discrete number of values.

Fig. 4.7 shows only the family of solutions with $i < 0.01$ deg. The number of solutions is considerably lower. Furthermore, some values of total time of flight are missing. This behaviour can be explained by looking at the resonances of the solutions.

If the analysis is restricted to the transfers taking less than 120 days and less than 150 m/s of total Δv, the solutions can be classified with respect to their resonances as in Fig. 4.8. The colour code corresponds to the number of revolutions of Ganymede around Jupiter between each consecutive pair of swing-bys. Since the spacecraft only performs one revolution around Jupiter in between two subsequent swing-bys, then the resonances are of type 1:*n* where *n* is the number revolutions of Ganymede.

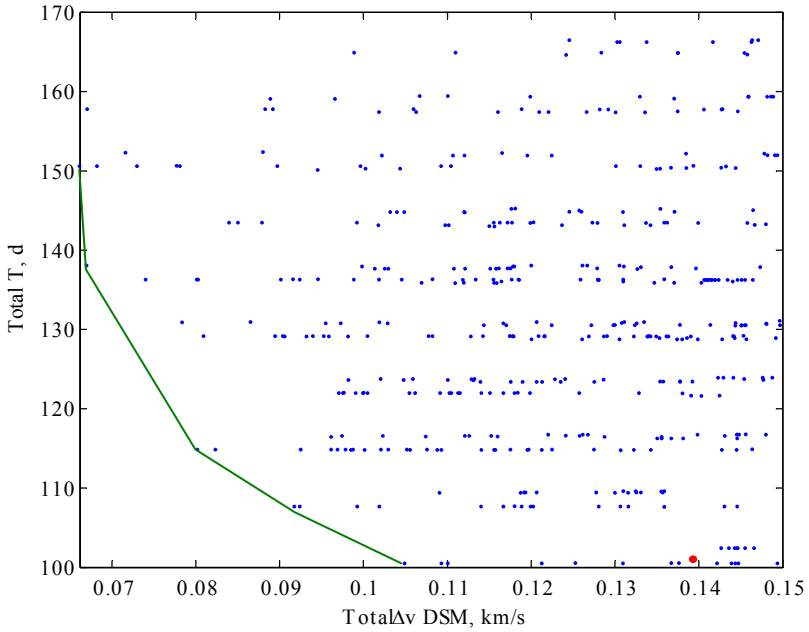


Fig. 4.6. Total Δ*v* and total time of flight for GGGG solutions with final inclination lower than 0.15 deg and total Δ*v* lower than 150 m/s. The red dot represents the baseline solution chosen by ESOC for the Laplace mission.

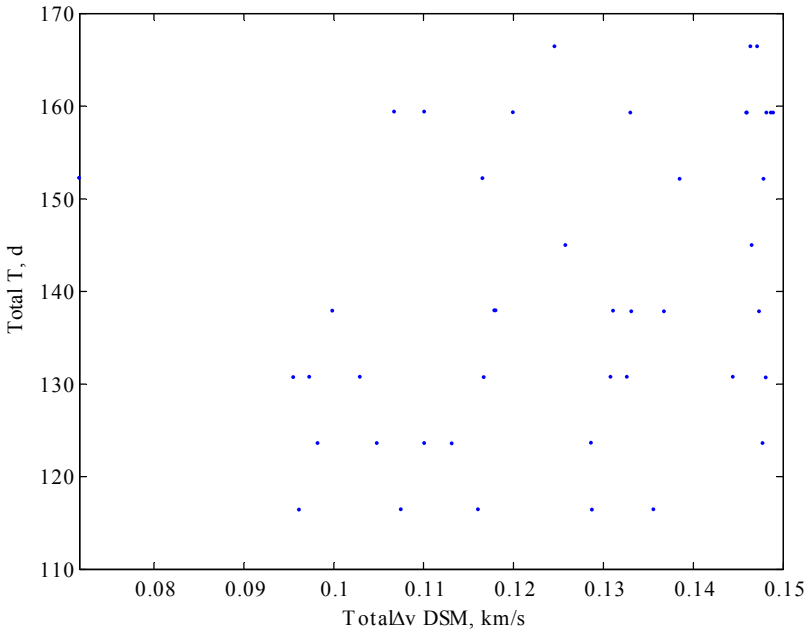


Fig. 4.7. **Total Δ*v* and total time of flight for GGGG solutions with final inclination lower than 0.01 deg and total Δ*v* lower than 150 m/s. This set of solutions corresponds to only one of the two families in Fig. 4.5.**

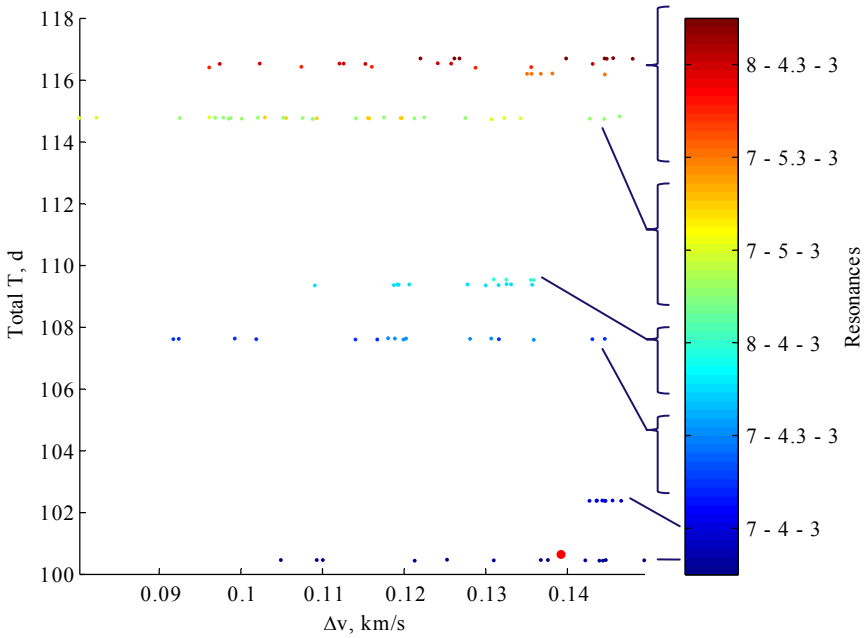


Fig. 4.8. **Total Δ*v* and total time of flight for the GGGG solutions with total time of flight lower than 120 days and total Δ*v* lower than 150 m/s. The red dot, in the 7 – 4 – 3 family, represents the baseline solution chosen by ESOC for the Laplace mission.**

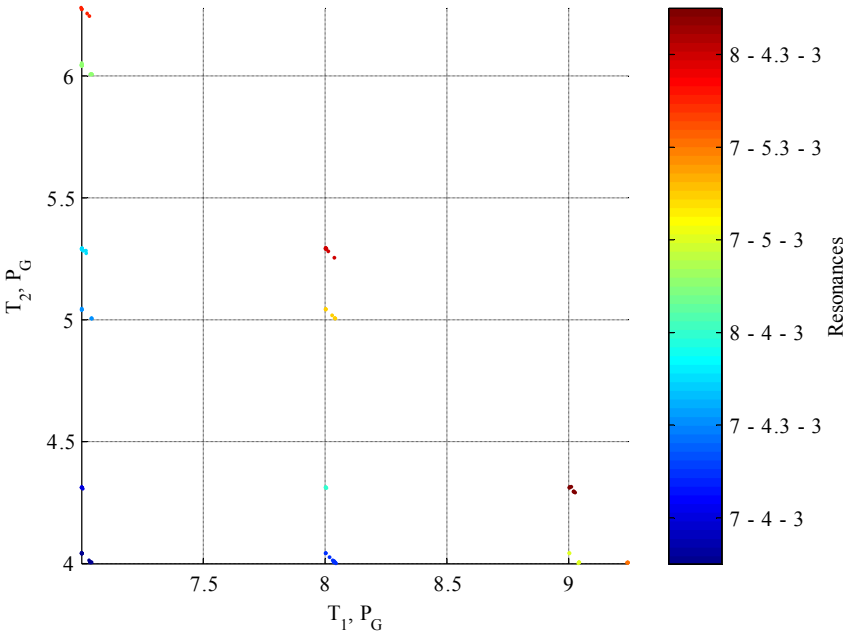


Fig. 4.9. Time of flight of the first and second leg, expressed in periods of Ganymede (P_G), for each solution. Time of flight of the third leg is $3P_G$ for all the solutions.

In Fig. 4.8, the number n is reported for each one of the three legs forming a triplet of numbers. Each family of solutions with the same time of flight has the same set of resonances for the three swing-bys and therefore the same triplet. Note that the last resonance in each family is 3, since this is the required target value.

If n takes only integer values within a triplet (for example 7 – 4 – 3, 8 – 4 – 3, 7 – 5 – 3), then the swing-by always occurs at the same position along the orbit of Ganymede. When a non-integer number of resonances is present in the triplet (e.g. 7 – 4.3 – 3), the spacecraft is changing the position of the swing-by, switching to the other orbital intersection. This also means that if a given family has an odd number of non-integer resonances, then the last Ganymede encounter occurs at a different point along the orbit.

Fig. 4.9 represents the same solutions of Fig. 4.8, with the same colours according to the resonances, in a plane with the first and the second resonance, being the third always equal to 3, for all the solutions.

An interesting aspect of this analysis is that solutions belonging to the same family of resonances have different costs. For example, Fig. 4.10 illustrates the trajectory of the cheapest and most expensive solution for the families 7 – 4 – 3, 8 – 4.3 – 3, and 7 – 4.3 – 3. The same figure shows also the Δv (in m/s) for each DSM. Cheap solutions perform a significant manoeuvre after the first swing-by, while the subsequent legs are quasi-ballistic. On the other hand, expensive solutions apply the major Δv after the second swing-by. Therefore, it is more convenient to change the v_∞ as soon as possible while the apocentre of the orbit is still high.

Fig. 4.11, Fig. 4.12 and Fig. 4.13 show the trajectory for the cheapest solution in 9 – 4 – 3, 11 – 5.2 – 3, 9.2 – 4.8 – 3 families respectively. Correspondingly, Table 4.5, Table 4.6 and Table 4.7 show the v_∞ at each swing-by, the Δv for each leg and the evolution of the orbital inclination following each event. The v_∞ is changed to the target value of 5 km/s with the first DSM while the inclination reaches its target value after the second or third swing-by.

The three different solutions also make clear the reason for the existence of two different values of inclinations (see again Fig. 4.5). Trajectories in the family 9 – 4 – 3 and 9.2 – 4.8 – 3, as well as all other families with an even number of non-integer resonances, can only achieve a minimum inclination of 0.12 deg, that is the inclination of Ganymede's orbit. On the other hand, solutions that are changing the swing-by position on the other intersection, can achieve a much smaller inclination, being that point closer to the line of nodes of Ganymede's orbit.

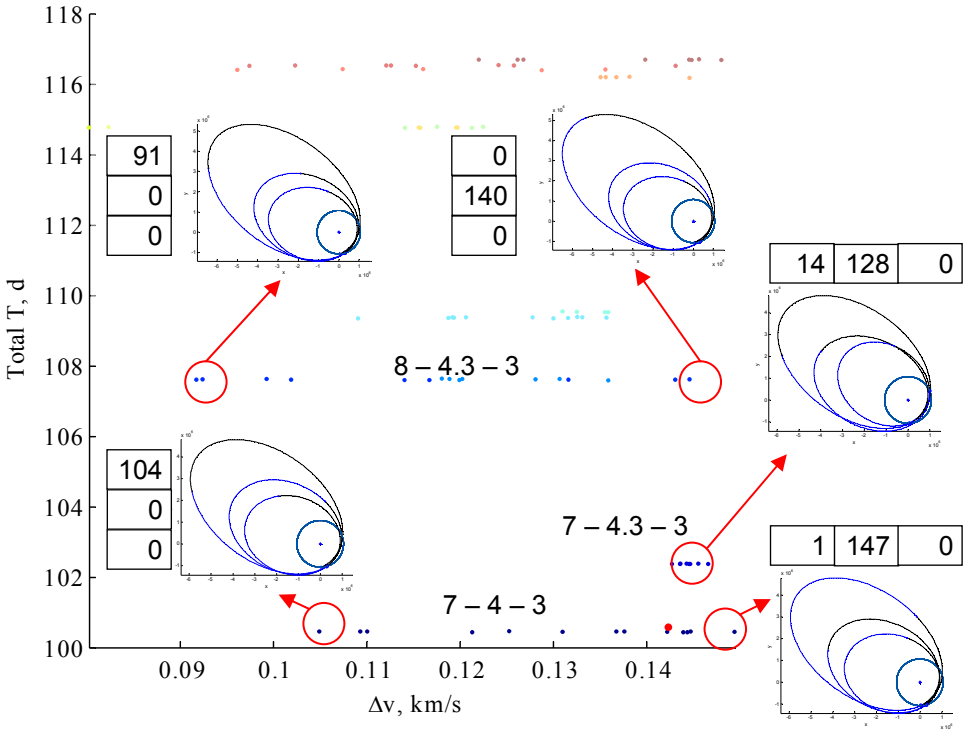


Fig. 4.10. Projection on the x-y plane of the cheapest and most expensive solution in three resonance families. The three numbers next to each plot show the magnitude of each DSM in the solution, in m/s.

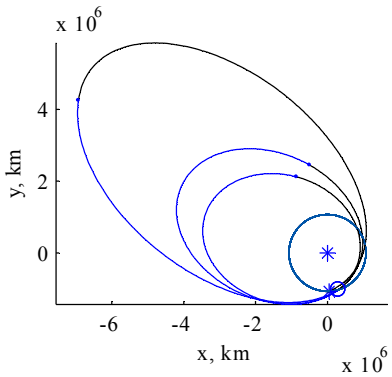


Fig. 4.11. Projection in the x-y plane of the cheapest 9 – 4 – 3 solution for the GGGG case.

Table 4.5. Evolution of Ganymede relative velocity and inclination, and applied Δ*v*, for the solution in Fig. 4.11.

	v_∞, km/s	Δv, km/s	i, deg
SB 1	6		3.1
DSM 1		78	3.8
SB 2	5		2.6
DSM 2		0	2.6
SB 3	5		0.1
DSM 3		1	0.1

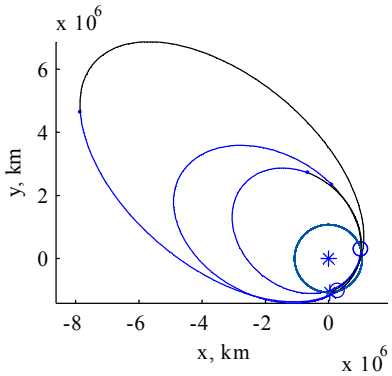


Fig. 4.12. Projection in the x-y plane of the cheapest 11 – 5.2 – 3 solution for the GGGG case.

Table 4.6. Evolution of Ganymede relative velocity and inclination, and applied Δ*v*, for the solution in Fig. 4.12.

	v_∞, km/s	Δv, km/s	i, deg
SB 1	6		2.3
DSM 1		67	2.8
SB 2	5		0.1
DSM 2		0	0.1
SB 3	5		0
DSM 3		0	0

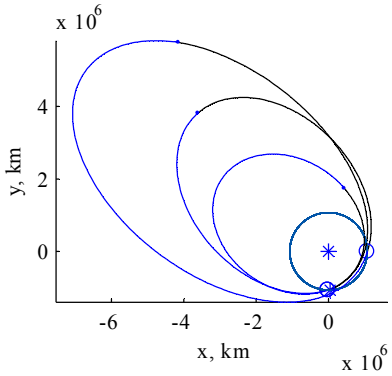


Fig. 4.13. Projection in the x-y plane of the cheapest 9.2 – 4.8 – 3 solution for the GGGG case.

Table 4.7. Evolution of Ganymede relative velocity and inclination, and applied Δ*v*, for the solution in Fig. 4.13.

	v_∞, km/s	Δv, km/s	i, deg
SB 1	6		2.6
DSM 1		139	1.9
SB 2	5		0.1
DSM 2		0	0.1
SB 3	5		0.1
DSM 3		0	0.1

The number of function evaluations and the time required to prune and optimise the complete problem is shown in Table 4.8.

In order to test the sensitivity to the target value of v_∞, a second analysis was run with $v_\infty = 4.5$ km/s, and the same value of target inclination (0 deg) and period (3 periods of Ganymede). The objective function and all the parameters were kept unchanged, except for u_l and u_u in A_i (see Table 4.9). This forces the v_∞ to be very close to 4.5 km/s after the third DSM.

Fig. 4.14 shows the Δv and total time of flight for the solutions with $i < 0.15$ deg, $\Delta v < 150$ m/s, and final v_∞ with respect to Ganymede between 4.48 and 4.52 km/s. As a reference only, the red dot represents the baseline solution (whose final v_∞ is 5 km/s). The figure shows that for a lower value of the v_∞, short transfer solutions do not exist anymore. There are solutions cheaper than the baseline but for longer flight times.

Table 4.8. Number of function evaluation and computational time for each level of the GGGG transfer problem.

Level	No. objective function evaluations	Time Intel	Time Sun
1	118,432	243 s	607 s
2	423,688	22 min	54 min
3	2,040,725	133 min	332 min

Table 4.9. Parameters u_l and u_u for the objective function at level 3.

Level l	$g_{v_\infty,l}(v_\infty)$	
	u_l, km/s	u_u, km/s
3	4.48	4.52

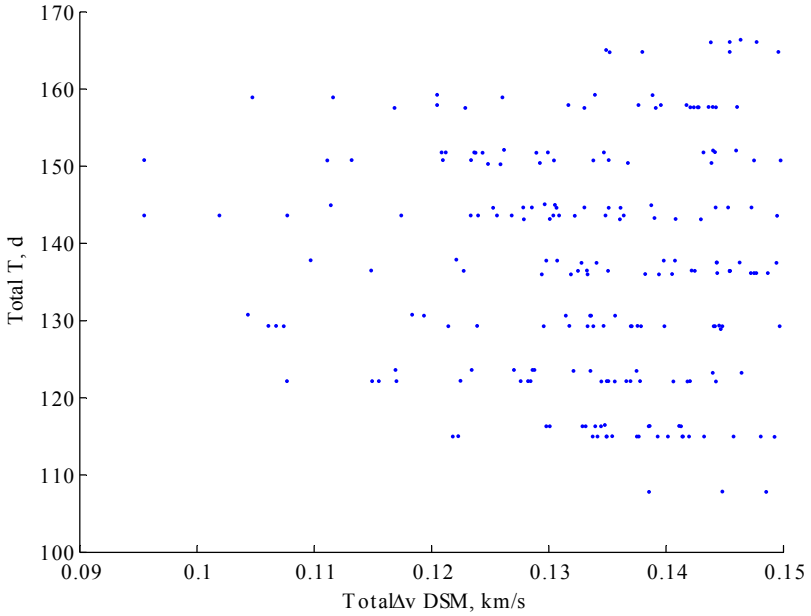


Fig. 4.14. Total Δv and total time of flight for the GGGG solutions with final inclination lower than 0.15 deg, total Δv lower than 150 m/s, and final relative velocity to Ganymede equal to 4.5 km/s. The red dot represents the baseline solution chosen by ESOC for the Laplace mission: note that the final relative velocity for this solution is 5 km/s.

4.2.2 Ganymede to Callisto Phase

This test case is about the design of the transfer from Ganymede (G) to Callisto (C), i.e. the GGA5-C phase of the Laplace mission. For this test case, it is assumed that the incoming conditions (velocity, time and position) at the fifth Ganymede swing-by are known, and represent the initial values for the MGA transfer to Callisto (see Table 4.10). These values were chosen to match the baseline solution provided by ESA for ease of comparison. The reference frame for this problem is the inertial Jupiter equatorial reference frame.

Table 4.10. Initial epoch and initial velocity relative to Ganymede for the GGA5-C transfer case.

t_0	9331.2989 d, MJD2000 (19 July 2025)
$\mathbf{v}_{\infty,r\theta h}$, km/s	[1.6234, 4.8384, -0.0001]

The aim of the transfer is to reach Callisto with a relative velocity as close as possible to 2 km/s, in magnitude, at minimum Δv cost. Only Ganymede and Callisto are considered. An additional requirement is to minimise the total time of flight to reduce mission operations cost. A further requirement would be to maintain the radius of the perijove as high as possible to avoid the belts of highly charged

particles surrounding Jupiter. This constraint was not considered in this analysis although it could have been used for further pruning of the solution space.

SEQUENCE SELECTION

As multiple celestial bodies are available, the selection of the sequence of swing-bys is a key element of the design process. The algorithm for generating and assessing the possible sequences was run considering possible swing-bys of G and C. The other parameters were set as following:

$$n_{sb,max} = 4$$
$$n_{rsb,max} = 2$$
$$n_{back,max} = 2$$
$$n_{backspacing,max} = 1$$

The list of radii of pericentre of the swing-bys was set to:

$$\mathbf{r}_p = \begin{bmatrix} 1.1 & 1.5 & 2 \end{bmatrix}$$

expressed in radii of the planet where the swing-by happens. The first swing-by of the sequence is at G because the initial conditions are given *at* G before its swing-by. The sequences found by the algorithm, and the *minimum* value of the v_∞ at arrival at C for each sequence, are shown in Table 4.11. The sequences in bold were selected as candidates for further study and optimisation with the incremental pruning

The reason for choosing the minimum value of v_∞ and not, for example, the one closest to the target of 2 km/s is that it is more likely that the full-model problem can reach the target of 2 km/s with low DSM Δv.

Table 4.11. Sequences, radii of pericentre of each swing-by and minimum achievable relative velocity at C. In bold, the sequences that were selected.

Sequence					r_p, R_P (direction)				v_∞ at C, km/s
G	C				1.1 (-)				3.78
G	**G**	**C**			**2 (-)**	**1.1 (-)**			**2.42**
G	C	C			1.1 (-)	2 (-)			3.78
G	**G**	**C**	**C**		**2 (-)**	**1.1 (-)**	**1.1 (+)**		**2.42**
G	**C**	**G**	**C**		**1.1 (-)**	**1.5 (+)**	**2 (-)**		**1.77**
G	C	C	C		1.1 (-)	2 (-)	2 (-)		3.78
G	G	C	G	C	1.1 (-)	2 (+)	2 (+)	1.1 (-)	2.43
G	C	G	G	C	2 (+)	2 (+)	1.1 (-)	1.1 (-)	2.60
G	**C**	**G**	**C**	**C**	**1.1 (-)**	**1.5 (+)**	**2 (-)**	**2 (+)**	**1.77**
G	C	C	G	C	2 (-)	2 (-)	1.1 (+)	1.1 (-)	1.95

INCREMENTAL SOLUTION

The trajectory starts with a resonant swing-by of Ganymede, thus the solution vector is defined as (see Eq. (2.19):

$$\mathbf{x} = \left[\gamma_1, r_{p,1}, \alpha_1, T_1, ..., \gamma_i, r_{p,i}, \alpha_i, T_i, ..., \gamma_{n_{legs}}, r_{p,n_{legs}}, \alpha_{n_{legs}}, T_{n_{legs}} \right],$$

where n_{legs} can be either 2, 3 or 4, depending on the length of the selected sequence.

The partial objective functions for each level (except the last one, the complete problem) were set to:

$$f_l = \sum_{i=1}^{l} \Delta v_i, \qquad l = 1 ... n_{legs} - 1 \tag{4.3}$$

The optimiser was stopped when all the Δv_i of each leg were below 100 m/s each. An analogous condition was set as a pruning threshold: a solution was considered feasible if all the Δv_i were below 100 m/s. This translates in the following pruning criteria and thresholds:

$$\Phi_l(\mathbf{x}_l) = \Delta v_i \le 100 \text{ m/s}, \quad i = 1,...,l, \quad l = 1,...,n_{legs} - 1 \tag{4.4}$$

Multi-Start optimisation coupled with Mean Shift clustering were used to prune the solution space.

The transfer from Ganymede to Callisto has multiple objectives and it would be natural to employ a multi-objective optimisation heuristic. However, the pruning process is based on the evaluation of a single objective per level. Therefore, rather than extending the incremental algorithm to deal with multi-objective problems, in this work it was decided to use one of the following four ways of tackling multiple criteria:

1. The objective function was casted into a weighed sum of the total Δv and $\left(v_\infty - 2 \text{ km/s}\right)^2$. The weight was kept fixed during the optimisation.
2. A constrained optimisation was set, with the objective of minimising the total Δv, under the constraint that $v_\infty < 4$ km/s. This approach is computationally more expensive that the single objective, unconstrained problem, because of the evaluation of the constraint function.
3. The total Δv is minimised without taking into account the v_∞ at all.
4. The objective function was defined as in point 1, but the weight was included in the solution vector, as an additional free parameter of the optimisation.

The four different approaches were applied to the last (third) level of the GCGC transfer case, after having pruned the search space at level 1 and 2 as before. The results found for the GCGC sequence have general validity and can be extended to the other sequences.

The total Δv, total time of flight and v_∞ at C, for all the solutions found by the four different approaches, are plotted in Fig. 4.15. The solutions can be grouped

into six families according to the total time of flight: plots in Fig. 4.16 represent each one of the first four families, those with shorter time of flight.

All four approaches generate the solution circled in red in Fig. 4.16, which is the baseline for the Laplace mission. The constrained and augmented approaches (green and black dots), yield results of similar quality but are computationally more expensive than the others. The approach with free v_∞ (blue dots) presents a high number of solutions with very high v_∞, as expected, meaning that computational time was spent looking for solutions which are not very interesting for this test case.

It was then decided to use an objective function weighing the Δv and the v_∞ in the following form:

$$f(\mathbf{x}) = \sum_i \Delta v_i + \beta \left(v_\infty - 2 \text{ km/s}\right)^2 \tag{4.5}$$

The value of β was set, through a rule of thumb, to 0.25, considering the order of magnitude of the two terms in objective function.

The following sub-sections will present the results of the incremental pruning process of the selected sequences and the solutions found following the subsequent optimisation of the residual search space.

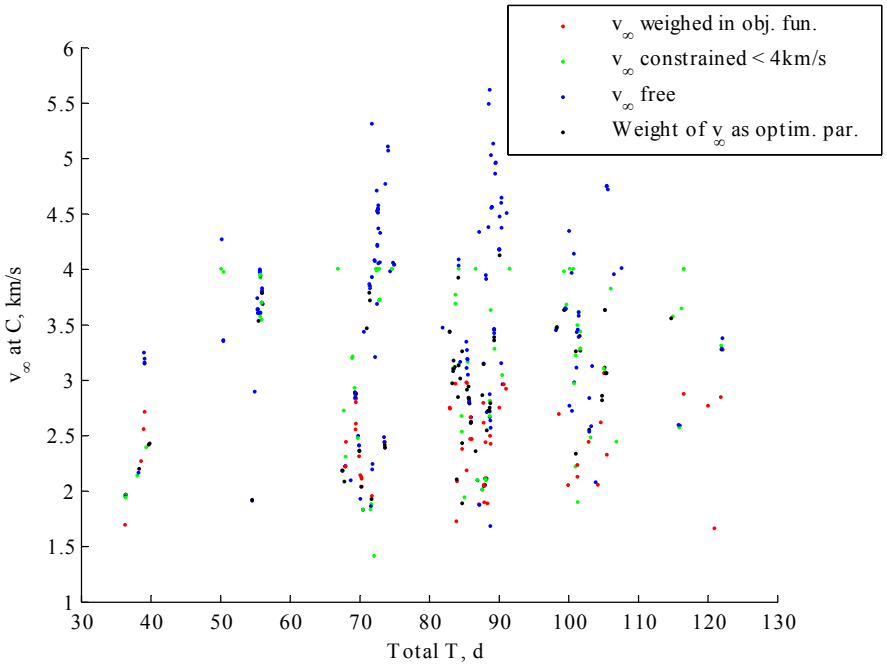


Fig. 4.15. Final relative velocity at C and total time of flight of the solutions found using the four different objective functions. The solutions resulted to be clustered into six families according to the total time of flight.

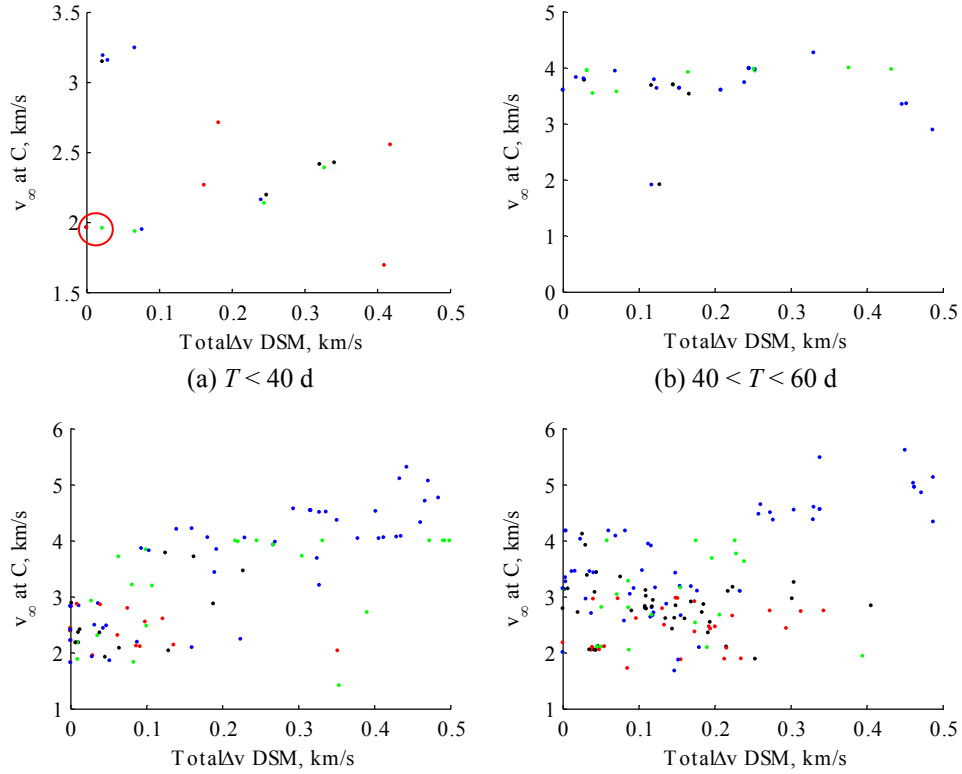


Fig. 4.16. Final relative velocity at C and total Δ*v* of the solutions found using the four different objective functions. Each plot refers to one family of solutions visible in Fig. 4.15, according to the total time of flight *T*.

RESULTS

In the following sections, we will present the results of the incremental pruning for each one of the planetary sequences highlighted in Table 4.11.

Sequence GCGC

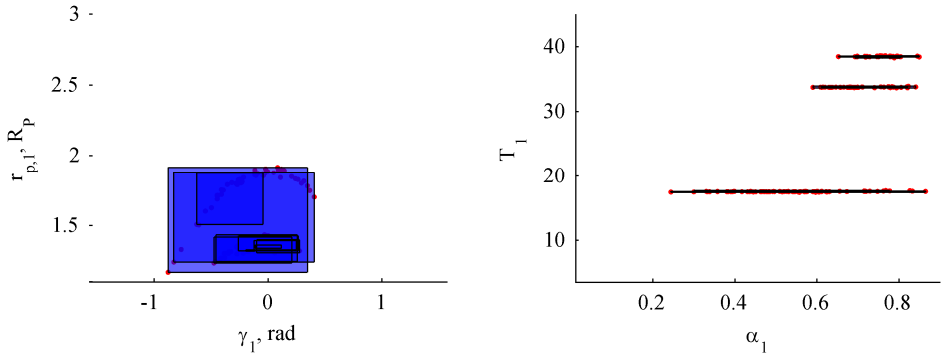
The global bounds for the problem were set as in Table 4.12. The bounds on γ were fixed according to the information on the deflection given by the energetic approach. Fig. 4.17 (a) and (b) show the feasible set at level 1 after pruning level 2: the solutions are clustered along the coordinate T_1 (see Fig. 4.17 (a)) and form two distinct structures in the space of the first swing-by $\left[\gamma_1, r_{p,1}\right]$ (see Fig. 4.17 (b)).

Two sets of solutions to the complete transfer problem are represented in Fig. 4.18 and Fig. 4.19. The red dots identify the solutions obtained with the incremental approach, while the blue solutions are the reference solutions computed with a systematic search with a planar trajectory model and tangential Δv of discrete magnitude located either at the pericentre or at the apocentre [118]. The blue solutions are not locally optimal, while the red solutions are locally optimal. Even

in this case, solutions can be grouped in families according to the total time of flight. For each family the red solutions have lower values of the total Δv compared to the blue solutions.

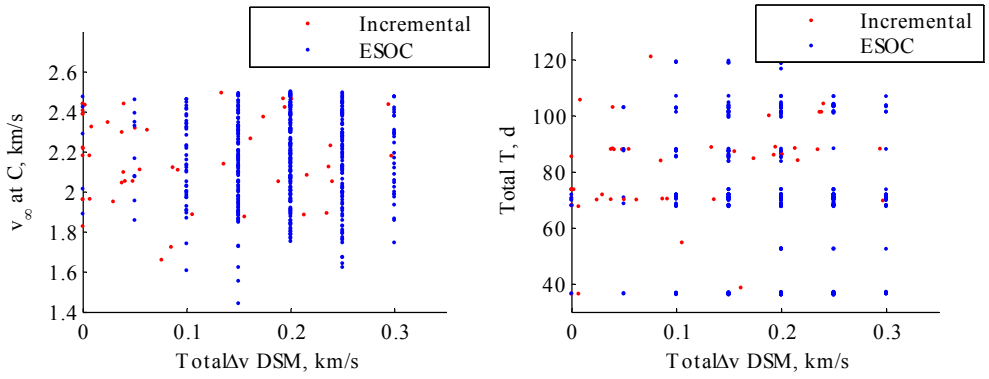
Table 4.12. Bounds for the GCGC transfer case.

Variable	LB	UB
T_i, d	3.5	45
$r_{p,i}$, R_P	1.1	3
α_i	0.01	0.9
γ_1, rad	$-\pi/2$	$\pi/2$
γ_2, rad	$\pi/2$	$3\pi/2$
γ_3, rad	$-\pi/2$	$\pi/2$



(a) Projection along γ_1, $r_{p,1}$ (b) Projection along α_1, T_1

Fig. 4.17. Projections of solutions and feasible regions at level 1 after pruning level 2.



(a) Δv vs. v_∞ (b) Δv vs. T

Fig. 4.18. GCGC sequence: comparison between ESOC (blue) and incremental (red) solutions. Only solutions with relative velocity at C lower than 2.5 km/s and Δv lower than 300 m/s are represented in the figure.

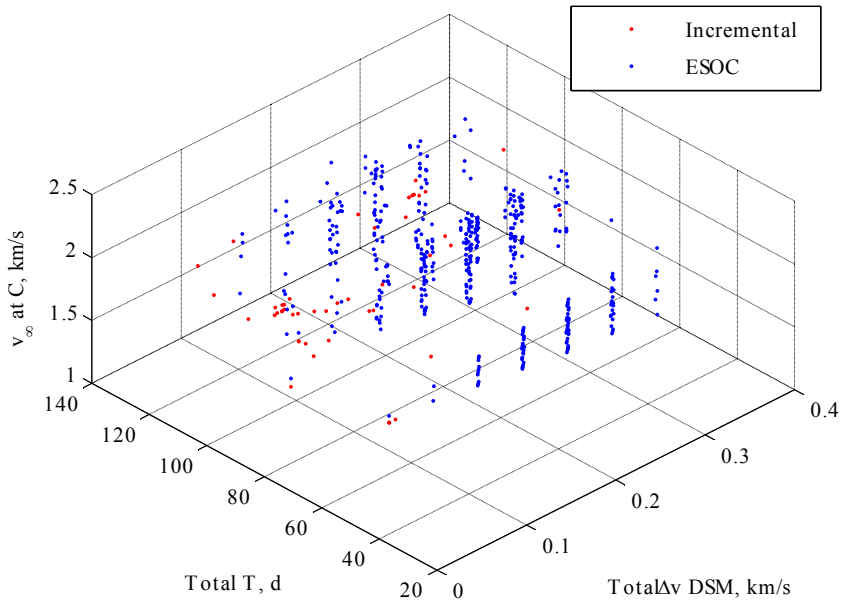


Fig. 4.19. GCGC sequence: comparison between ESOC (blue) and incremental (red) solutions. Only solutions with relative velocity at C lower than 2.5 km/s and Δv lower than 300 m/s are represented in the figure.

Table 4.13. Characteristics of the baseline GCGC solution.

T_1, d	17.5
T_2, d	13.8
T_3, d	5.1
Total Δv, m/s	0
Total time of flight, d	36.4
v_∞ at C, km/s	1.96

As stated before, the solution chosen as a mission baseline by ESA has been also found by the incremental pruning and subsequent optimisation. Its characteristics and trajectory are represented in Table 4.13 and Fig. 4.20 respectively.

The number of function evaluations required to obtain these solutions, level by level, and the corresponding computational time, are shown in Table 4.14.

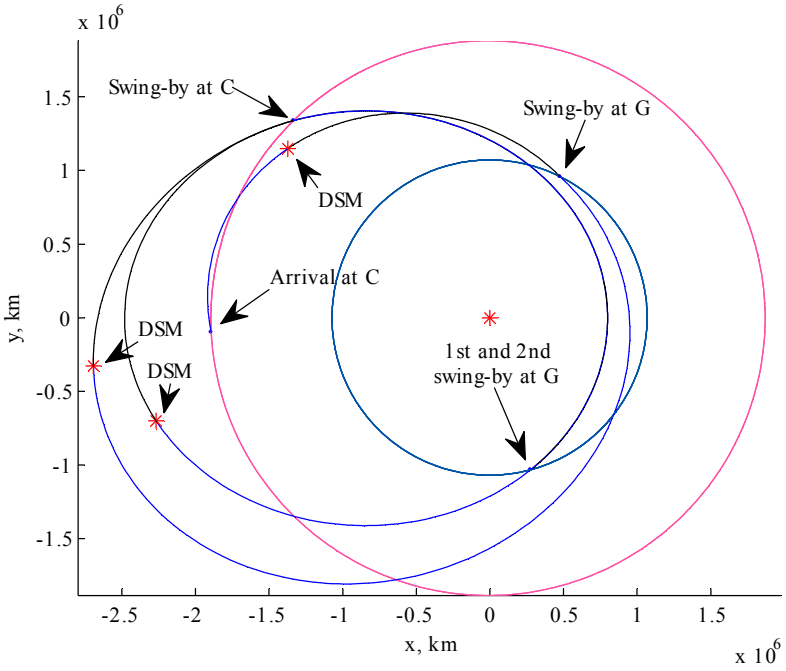


Fig. 4.20. x-y projection of the baseline GCGC solution.

Table 4.14. Number of function evaluation and computational time for each level of the GCGC transfer problem.

Level	No. objective function evaluations	Time Intel	Time Sun
1	61,082	118 s	295 s
2	418,150	21 min	53 min
3	770,508	54 min	135 min

Sequence GGCC

The bounds used for this sequence are in Table 4.15. The GGCC sequence was tackled in the same way of the GCGC. Results in terms of relative final velocity, Δv and total transfer time are in Fig. 4.21 and Fig. 4.22 (red dots). Again, a comparison with a systematic approach (blue dots) is shown.

Table 4.15. Bounds for the GGCC transfer case.

Variable	LB	UB
T_i, d	3.5	45
$r_{p,i}$, R_P	1.1	3
α_i	0.01	0.9
γ_1, rad	$-\pi/2$	$\pi/2$
γ_2, rad	$-\pi/2$	$\pi/2$
γ_3, rad	$\pi/2$	$3\pi/2$

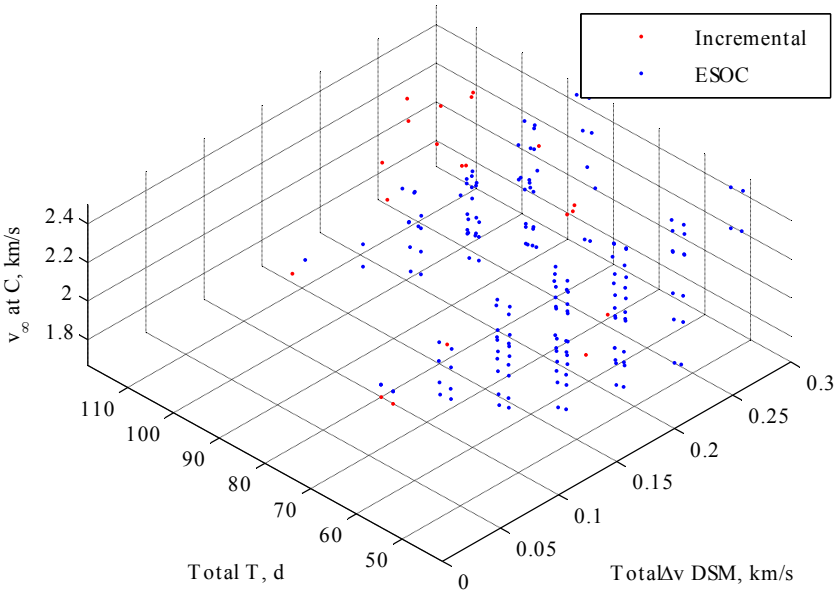
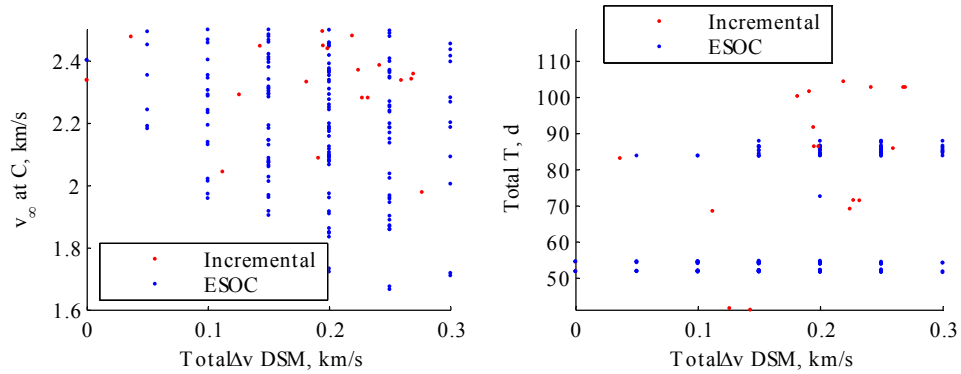


Fig. 4.21. GGCC sequence: comparison between ESOC (blue) and incremental (red) solutions. Only solutions with relative velocity at C lower than 2.5 km/s and Δv lower than 300 m/s are represented.



(a) Δv vs. v_∞ (b) Δv vs. T

Fig. 4.22. Two projections of Fig. 4.21.

Sequence GGC

This sequence generates an incremental problem with only 2 levels. The associated bounds are in Table 4.16, and solutions found are in Fig. 4.23. Among all the solutions of this case, a quite promising one was found: it is circled in Fig. 4.24. This solution arrives at C with a very low v_∞, compared to all the other solutions, and a low total Δv. Furthermore, it was noted that T_2 was exactly 3.5 days, meaning that the solution hit the lower bound during the optimisation. Removing this constraint and re-starting the optimiser led to a ballistic solution, although with a slightly higher v_∞. The characteristics of this new solution are summarised in Table 4.17, and its trajectory in Fig. 4.25.

Table 4.16. Bounds for the GGC transfer case.

Variable	LB	UB
T_i, d	3.5	45
$r_{p,i}$, R_P	1.1	3
α_i	0.01	0.9
γ_1, rad	$-\pi/2$	$3\pi/2$
γ_2, rad	$-\pi/2$	$3\pi/2$

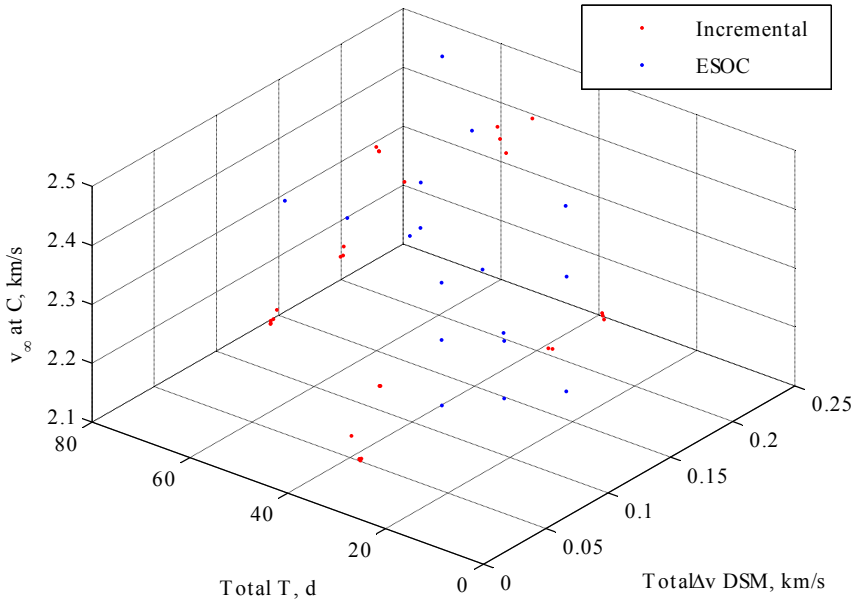
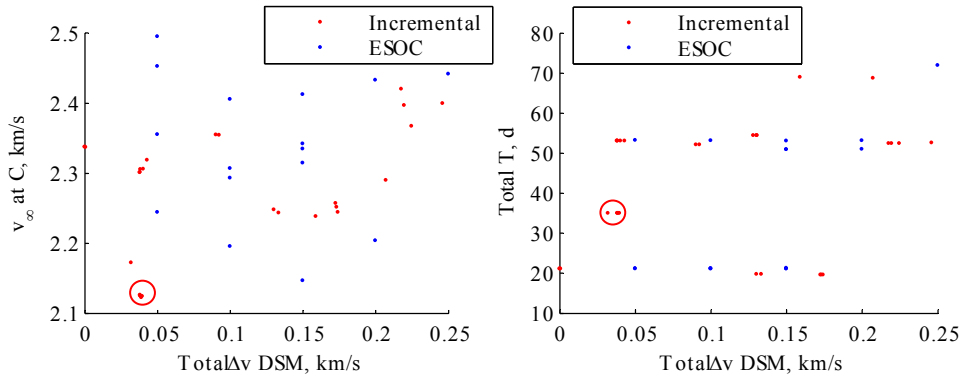


Fig. 4.23. GGC sequence: comparison between ESOC (blue) and incremental (red) solutions. Only solutions with relative velocity at C lower than 2.5 km/s and Δv lower than 250 m/s are represented in the figure.



(a) Δv vs. v_∞ (b) Δv vs. T

Fig. 4.24. Two projections of Fig. 4.23. The red circle identifies the promising solutions with an active constraint on time of flight on the second leg.

Table 4.17. Characteristics of the re-optimised GGC solution.

T_1, d	31.4
T_2, d	3.4
Total Δv, m/s	0
Total time of flight, d	34.8
v_∞ at C, km/s	2.17

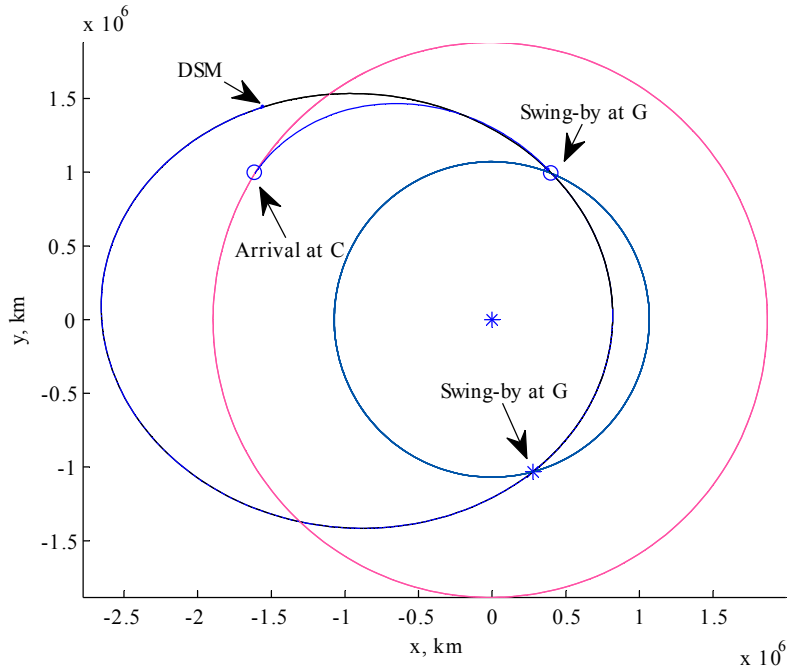


Fig. 4.25. x-y projection of the GGC solution after re-optimisation with a wider lower bound on time of flight for the second leg.

Table 4.18. Some examples of optimal GGC solutions.

Δv , m/s	v_∞ at C, km/s
18	2.14
45	2.09
59	2.07
68	2.06

The second leg has a very short time of flight of 3.4 days. Such a short leg represents an issue for two reasons. The first is the available time to operate the spacecraft from ground. The second is the effectiveness of the DSM at changing the terminal relative velocity with respect to Callisto. This means that the magnitude of the DSM can become approximately equal to the gain in relative velocity at C. A set of alternative solutions, with different v_∞ at C, were found, by changing the weight β in the objective function. The achievable v_∞, together with the cost of the (optimised) transfer is shown in Table 4.18.

Sequence GCGCC

This case has 4 levels to be pruned. The time of flight for each leg was upper limited to 24 days to limit the total transfer time. Bounds are presented in Table 4.19, and the resulting solutions are shown in Fig. 4.26 and Fig. 4.27.

Table 4.19. Bounds for the GCGCC transfer case.

Variable	LB	UB
T_i, d	3.5	24
$r_{p,i}$, R_P	1.1	10
α_i	0.01	0.9
γ_i, rad	$-\pi/2$	$3\pi/2$

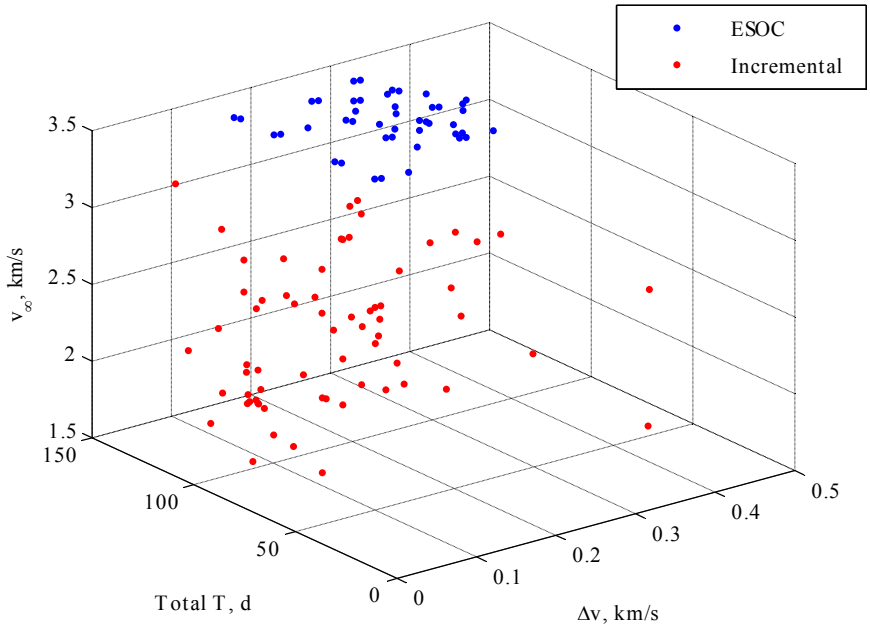
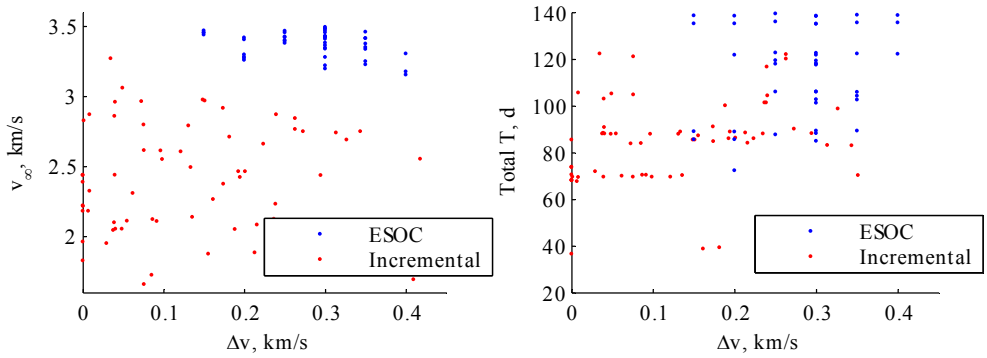


Fig. 4.26. GCGCC sequence: comparison between ESOC (blue) and incremental (red) solutions. Only solutions with relative velocity at C lower than 3.5 km/s and Δv lower than 450 m/s are represented in the figure.



(a) Δv vs. v_∞ (b) Δv vs. T

Fig. 4.27. Two projections of Fig. 4.26.

Analysis of Solution Families for Sequence GCGC

In this section, the solutions for the GCGC sequence (taken as a representative) will be classified according to two criteria. The first one makes use of the Tisserand plane (see Appendix E). Then, solutions will be classified total Δv, total time of flight, and final relative velocity.

The iso-v_∞ lines for Ganymede and Callisto are represented in Fig. 4.28, as well as the starting condition of the spacecraft (blue star). By means of a graphical analysis of the Tisserand plane, we can deduce that the spacecraft will perform one swing-by of Ganymede, changing mainly the period of its orbit, then one of Callisto, changing mainly the radius of pericentre of its orbit, and then the last swing-by of Ganymede, again changing mainly the period.

All the trajectories can be grouped into 4 types, according to the path they follow in the Tisserand plane. The main properties of each type are summarised in Table 4.20. Despite the sequence includes three swing-bys, only the first two (GC) will be included in this classification: in fact, the last swing-by (G) reduces the period of the orbit in all the cases.

The four plots in Fig. 4.29 show the optimal solutions for the GCGC transfer case: each plot refers to a different family according to the total time of flight; the colour of each point refers to the type of solution in the Tisserand plane, as in Fig. 4.28.

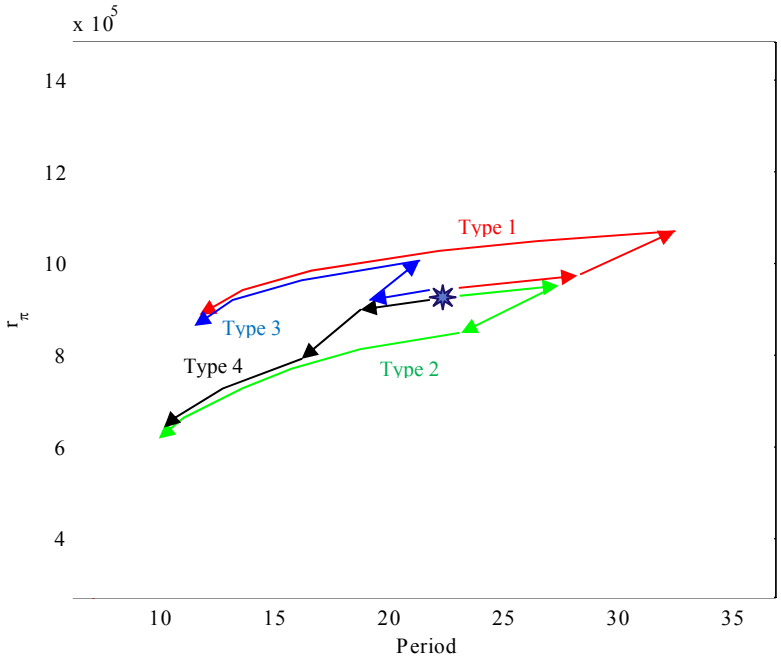
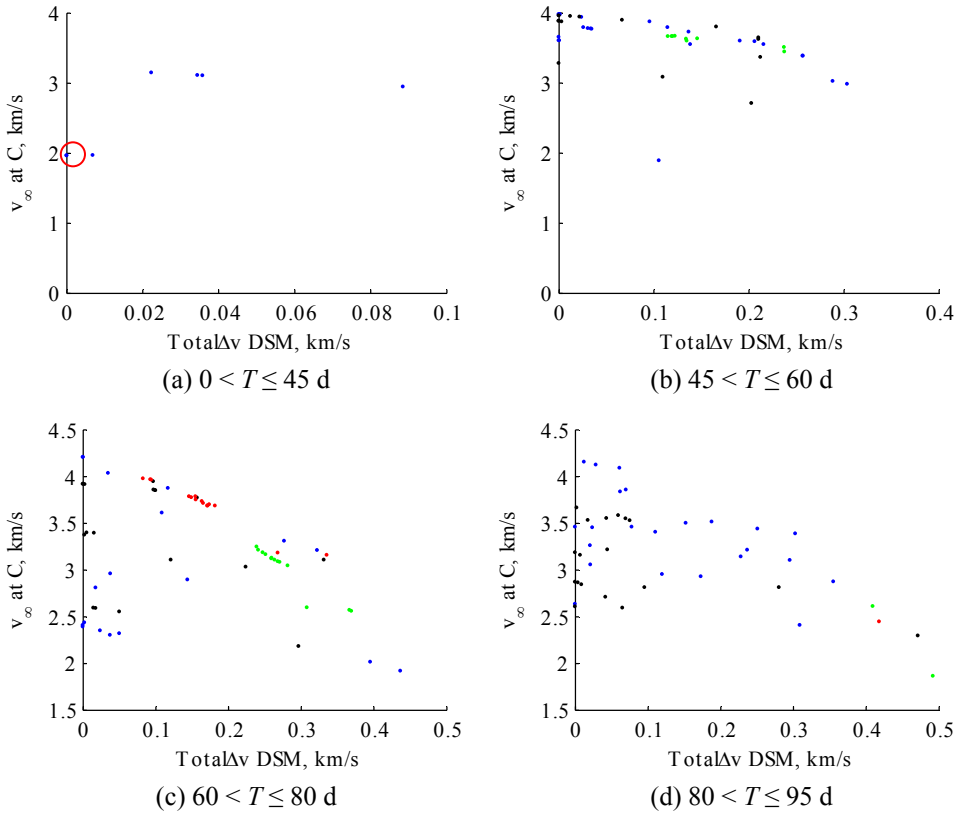


Fig. 4.28. Representation of the four types of solutions on the Tisserand plane. The blue star represents the initial orbit of the spacecraft, right before the first G swing-by. The target is to reach C on the 2 km/s iso-v_∞ line.

Table 4.20. Classification of the solutions according to the change of orbital period due to first swing-by of G and the change in radius of pericentre due to the first swing-by of C. The corresponding path in the Tisserand plane for each type is shown in Fig. 4.28.

Type	G (1) ΔP	C (1) Δr_π
• 1	> 0	> 0
• 2	> 0	< 0
• 3	< 0	> 0
• 4	< 0	< 0



(a) $0 < T \leq 45$ d (b) $45 < T \leq 60$ d

(c) $60 < T \leq 80$ d (d) $80 < T \leq 95$ d

Fig. 4.29. Representation of the four types of solutions to the GCGC on the v_∞-Δv plane. The red circle in (a) represents the GCGC baseline solution.

By looking at the position of the solutions of each type, we note that most of type 1 (red) solutions are in the 60~80 day family, and they are not very good concerning both the v_∞ and the Δv.

Type 2 solutions are mostly clustered in the 60~80 day family, having a quite high Δv, and in the 45~60 day family, having a very high v_∞.

Type 3 and type 4 solutions have considerably different Δv, v_∞, and total time of flight, so it is difficult to clearly identify clusters in this space. Both types contain

good solutions according to Δv and v_∞, but the short solutions (under 45 days) are all of type 3. Type 3 is also the baseline solution (highlighted with a red circle in top left plot of Fig. 4.29).

We can conclude that only the paths of type 1 and type 2 in the Tisserand plane have similar features: paths of type 3 and 4 in Tisserand plane generate a multitude of solutions with very different characteristics. Unfortunately, the most interesting solutions belong to these latter two types. We can then say that for this kind of problems, the Tisserand plane can provide a hint to find good solutions, but it is in no way a mean to uniquely identify good solutions. Additional tools are needed, that consider a more complete trajectory model.

Following the analysis with the Tisserand plane, we would like to investigate the similarities of solutions which have comparable parameters of merit (namely total Δv, time of flight and final relative velocity): in other words, we would like to see whether solutions which are comparable with regards to certain final conditions also have a similar trajectory in the space and/or a similar representation in the Tisserand plane. The important point is that, if two solutions have very close parameters of merit, and also similar trajectory, then the solutions are overall similar, and can be considered as one. On the other hand, if two solutions have similar parameters of merit, but different trajectories, then a trade-off should be done, based on other considerations, for ranking them and deciding which one (if any) is better.

We showed before (see Fig. 4.15) that all the solutions for the GCGC sequence are grouped into ideal "families" according to their total time of flight. We can use this consideration to base a classification of the solutions: we want to investigate whether solutions with similar total time of flight have similar trajectory and/or path in the Tisserand plane.

The first step, then, is to cluster the solutions into families according to their time of flight. This was done using Mean Shift, the same cluster algorithm used in the incremental method. The bandwidth was tuned such to obtain a reasonable number of clusters. Resulting clusters are shown in Fig. 4.30, which shows the space of the considered parameters of merit, and Fig. 4.31, which shows two projections of the previous figure. It is clear that the driving parameter for clustering is the time of flight: even if solutions which have similar time of flight, but considerably different other parameters, are not grouped together.

We consider only some of all these clusters, laying close one another, for further study. Fig. 4.32, which is a magnification of Fig. 4.30, highlights these clusters. For each one of the considered clusters, all the solutions within the cluster will be plotted in the same graph, to have a quick look of their similarities. Both the trajectory and the Tisserand plot will be assessed. If the solutions are overlapping one another, then they can be considered similar, and further investigation can be done. The purpose of the figures, then, is not to distinguish every single trajectory or path, but only to have an idea on their overlapping in either plot.

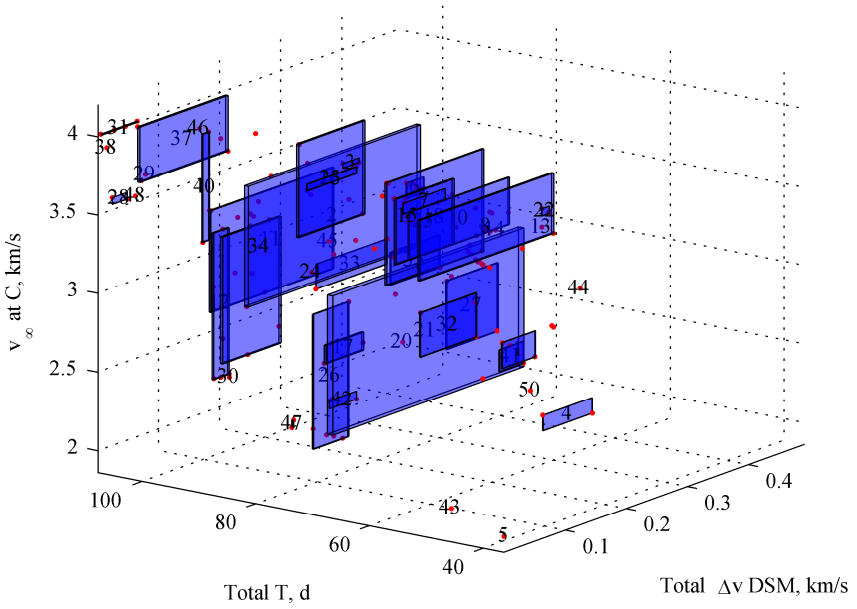
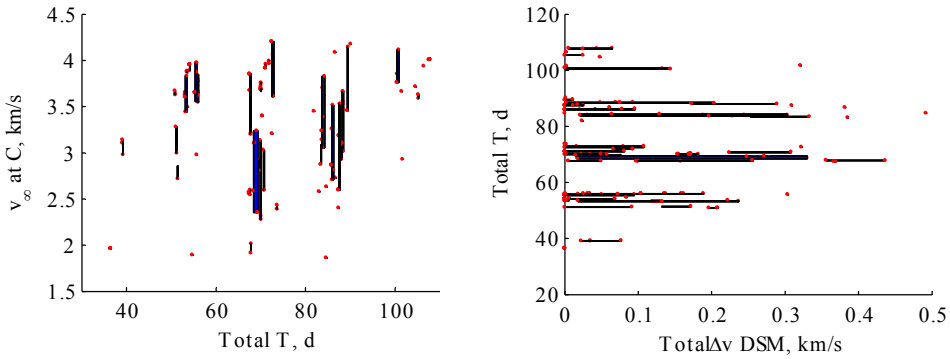


Fig. 4.30. Solutions clustered according to the total transfer time, in the space Δv-T-v_∞.



(a) T vs. v_∞ (b) Δv vs. T

Fig. 4.31. Two projections of Fig. 4.30.

The analysis is presented in the following series of figures (from Fig. 4.33 to Fig. 4.36), each one made of two plots. Each figure refers to one cluster of those numbered in Fig. 4.30, Fig. 4.31 and Fig. 4.32. In these figures, plots (a) represent the x-y projection of the trajectories corresponding to all the solutions in the cluster the figure refers to. Plots (b) are the Tisserand graphs of all the solutions in the same cluster. So both plots represent, in a different way, the same set of solutions, superimposed in the same picture.

In the trajectory plot, arcs in black are propagated (so before the DSM), while arcs in blue are Lambert's (so after the DSM). Each path in the Tisserand plane,

instead, is made of black curves and light blue lines. The line is black if the change in orbital parameters is due to a swing-by (and thus along an iso-v_∞ line), while displacements on the Tisserand plane due to a DSM are represented in light blue.

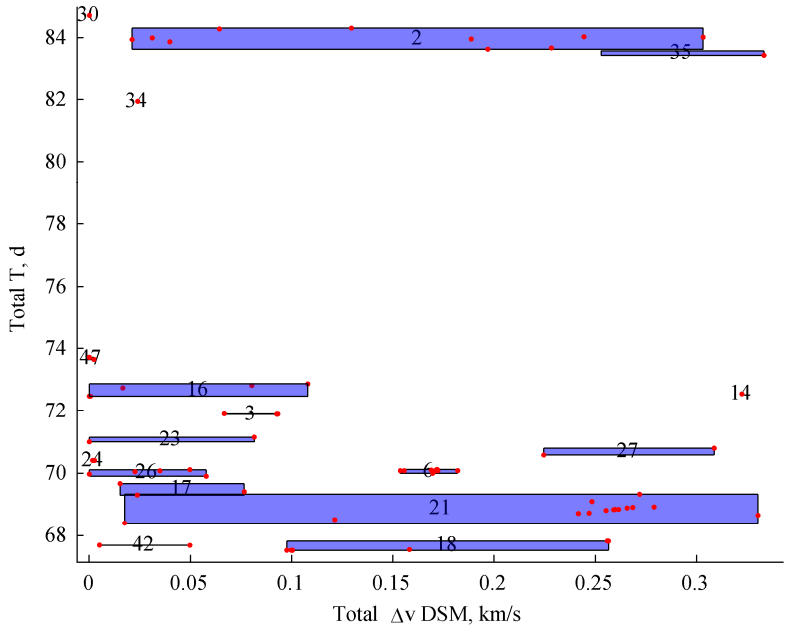
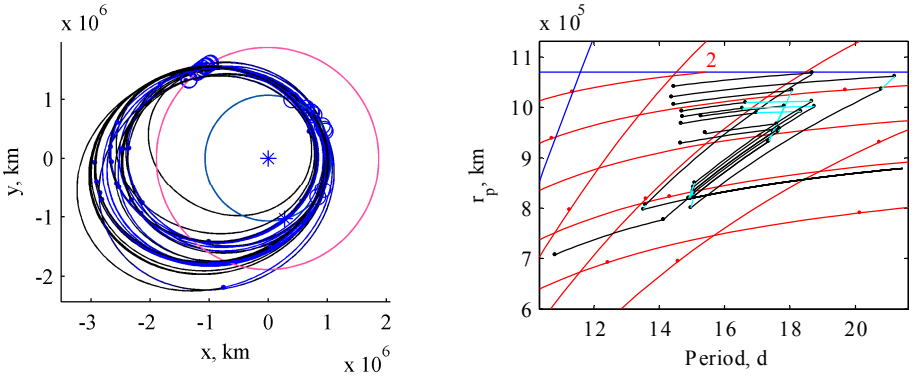
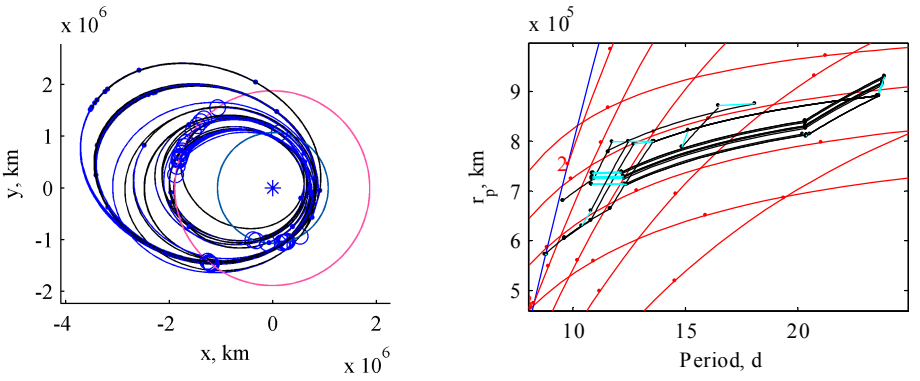


Fig. 4.32. Magnification of an area of Fig. 4.31 (b) to better visualise the clusters which will be considered.



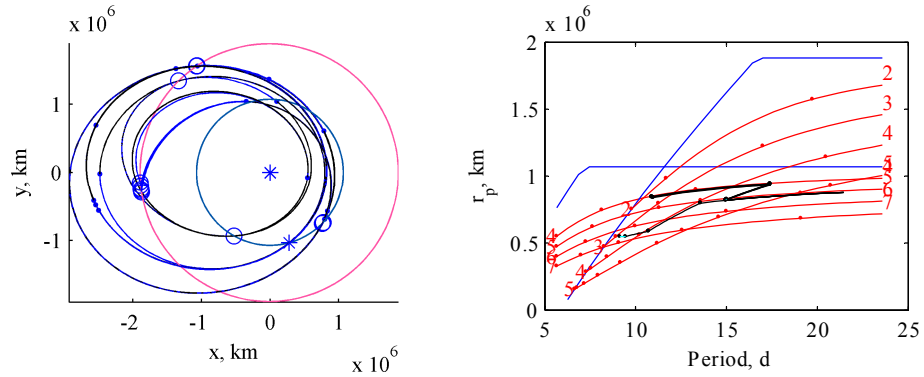
(a) Trajectory (b) Tisserand plane

Fig. 4.33. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 2.



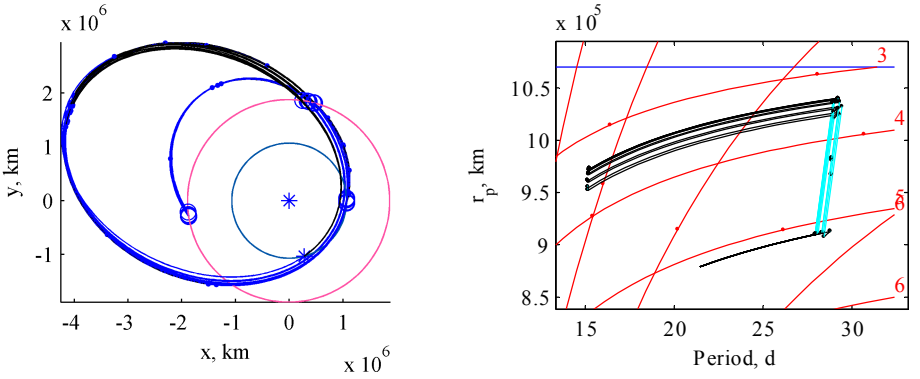
(a) Trajectory (b) Tisserand plane

Fig. 4.34. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 21.



(a) Trajectory (b) Tisserand plane

Fig. 4.35. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 26.



(a) Trajectory (b) Tisserand plane

Fig. 4.36. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 6.

Fig. 4.33 and Fig. 4.34 refer to clusters 2 and 21 respectively. These clusters include solutions which have similar time of flight, but are considerably different with respect to Δv and v_∞. As it is noticeable from the figures, the solutions are quite different, with respect to both the trajectory and the Tisserand plane: no overlapping can be spotted. In cluster 26 (Fig. 4.35), instead, solutions are partially overlapped, while in cluster 6 (Fig. 4.36) the overlapping is complete. These clusters, as opposed to the previous two, are very small: all the solutions they contain have very similar time of flight, Δv and v_∞.

We can conclude that different solutions exist with very similar time of flight, and thus the time of flight alone is not representative of a family of solutions. It seems instead that solutions with similar parameters of merit – all of them – have also similar trajectory and Tisserand graph.

In order to verify this last statement, solutions have been re-clustered according to their vicinity in the space total time of flight, v_∞, Δv. This time, the time of flight was not favoured in any way with respect to the other two parameters of merit. The resulting clusters, which are more and smaller than before, are shown in Fig. 4.37 and Fig. 4.38. Each cluster now includes solutions which have all three parameters of merit very similar one another.

The clusters considered for this second analysis are those appearing in Fig. 4.39, which is a detail of Fig. 4.37.

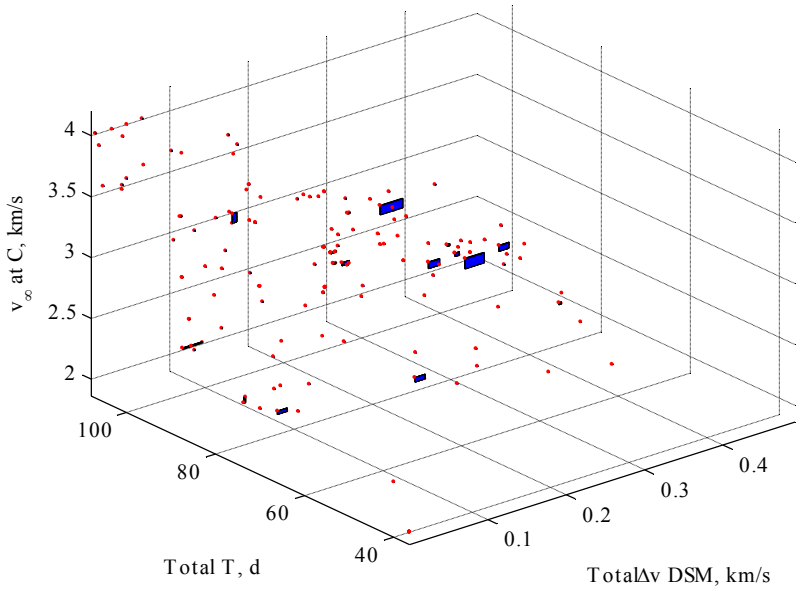
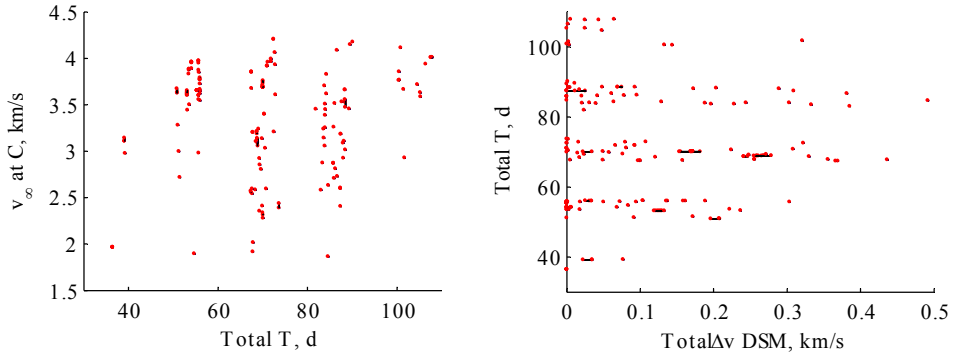
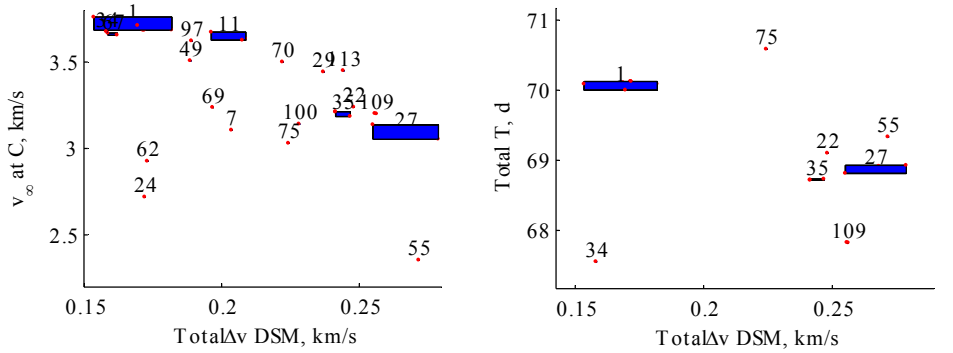


Fig. 4.37. Tight clustering of the solutions in the space Δv-T-v_∞. Most of the clusters include one solution only.



(a) T vs. v_∞ (b) Δv vs. T

Fig. 4.38. Two projections of Fig. 4.37.



(a) Δv vs. v_∞ (b) Δv vs. T

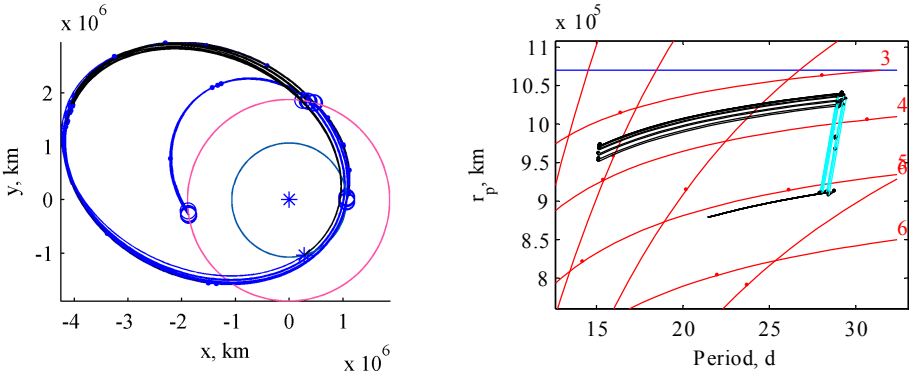
Fig. 4.39. Magnification of an area of Fig. 4.38 to better visualise the clusters which will be considered.

Fig. 4.40 to Fig. 4.43 are the x-y projections and Tisserand graphs of the solutions in each cluster. Some of the solutions are completely overlapped in both representations (clusters 1 and 27, corresponding to Fig. 4.40 and Fig. 4.41). In some other clusters, however, although the solutions have very similar parameters of merit, they have considerably different trajectories (clusters 22 and 109, corresponding to Fig. 4.42 and Fig. 4.43).

In particular, the case of cluster 109 is relevant: as it is clear from the trajectory x-y projection, the orbits of each leg are the same for the two solutions which belong to this cluster. Nevertheless, the position of the swing-bys along the orbits are different, and this leads to a completely different Tisserand paths. In other words, in this case, the Tisserand graph helps highlighting the completely different nature of these two solutions, despite their similarity in the x-y projection of the trajectory.

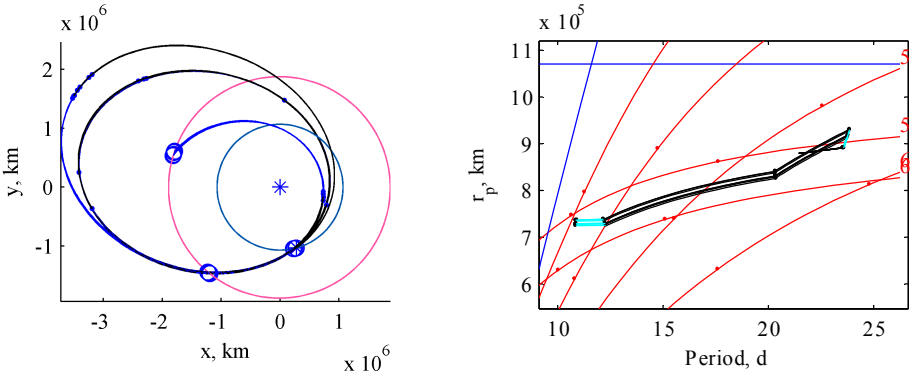
To conclude this part of the study, we can state that rather different solutions can exist, despite they have very similar parameters of merit (being Δv, final relative velocity v_∞ and total time of flight). So in other words the parameters of

merit are not enough to distinguish and classify the solutions of a MGA transfer, but other parameters should be considered.



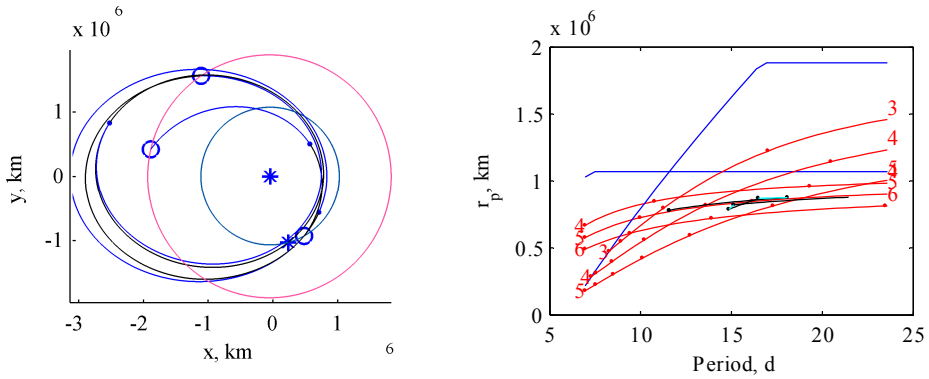
(a) Trajectory (b) Tisserand plane

Fig. 4.40. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 1.



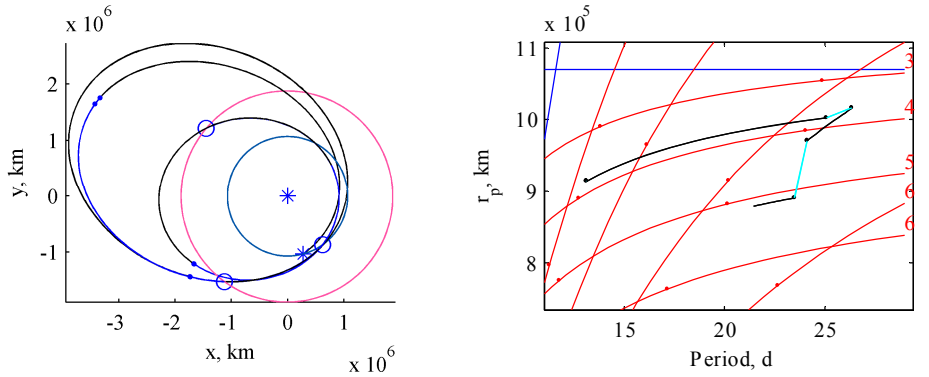
(a) Trajectory (b) Tisserand plane

Fig. 4.41. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 27.



(a) Trajectory (b) Tisserand plane

Fig. 4.42. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 22.



(a) Trajectory (b) Tisserand plane

Fig. 4.43. Projection of the trajectory and corresponding path in the Tisserand plane for the solutions in cluster 109.

45~60 d family

It has been already stressed how the solutions for the GCGC case could be grouped into families according to their total time of flight. However, if only solutions with a low final relative velocity (say below 3.5 km/s) are considered (Fig. 4.44), then it appears that for each family, there exist a ballistic solution, but for the family having the total time of flight between 45 and 60 days. The cheapest solution in this family required about 100 m/s of DSMs. The ballistic solutions and the cheapest solution in 45~60 day range are circled in red and green respectively in Fig. 4.44.

It is well known that DSMs can be exploited to change the v_∞ with respect to a given body. For this family, instead, it seems that the DSM is the only way to get a solution, regardless the value of v_∞ achieved. Now, the question is whether a ballistic solution exists in this family, if the constraint on the radius of the pericentre is removed. In other words, we would like to understand if it is possible to have a ballistic solution, when the swing-bys can provide an arbitrary deflection of the

relative velocity vector: in this case, the DSMs would be used to compensate for a lack of power in the swing-bys.

To answer these questions, note first that the radii of pericentre of the cheapest solution in the family are 1.38, 2.72 and 1.19 respectively. Since the lower bound was set to 1.1, the constraints are not active. This means that the solution remains a local minimum even using wider bounds for the radius of pericentre, or in other words that this solution would not benefit from a swing-by with a lower minimum radius. Other searches with wider constraints on r_p did not find to any better solution.

We can then conclude that for this family of solutions, DSMs are essential to reach the target with a sufficiently low value of v_∞, and this solution cannot be represented by a corresponding ballistic one. Further studies highlighted that the DSM changed slightly the period of the orbit, and thus allowing the phasing problem to be solved. The same change of period was not obtainable through a swing-by.

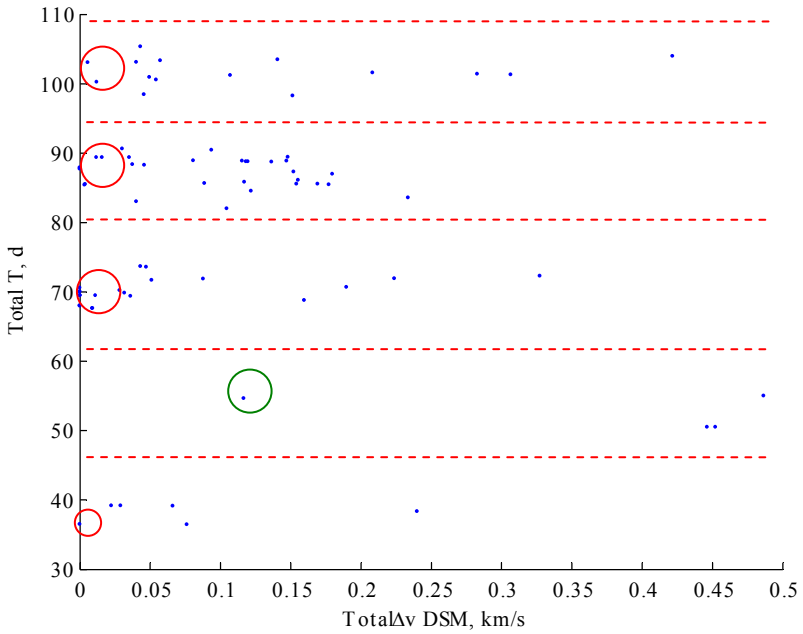


Fig. 4.44. GCGC solutions with final relative velocity lower than 3.5 km/s. There is no ballistic solution for a total time of flight of 45~60 days. The red circles identify the ballistic solutions. The green circle highlights the solution which was re-optimised.

4.3 BepiColombo Mission

This section presents the design of the chemical propulsion option for the BepiColombo mission [31]. The whole trajectory is made of two conceptually different parts: an MGA transfer from the Earth to Mercury and a resonant sequence of Δv-GAMs with Mercury to lower the final relative approach velocity to the planet. The two parts will be designed separately and the result compared to the baseline solution produced by ESA.

4.3.1 Earth to Mercury Transfer Phase

The problem is to find a suitable MGA trajectory to transfer the spacecraft from the sphere of influence of the Earth to Mercury with a launch in the interval [4500, 5500] MJD2000 (i.e. between April 2012 and January 2015). The sequence of swing-bys is free but only Earth, Venus and Mercury can be used. The relative velocity at Mercury is constrained to be below 5.8 km/s, and the total Δv and time of the transfer T need to be minimal. The reference frame is the Sun centred inertial ecliptic at epoch.

SEQUENCE SELECTION

The parameters for the generation of the sequence were set as follows:

$$n_{sb,max} = 3$$
$$n_{rsb,max} = 2$$
$$n_{back,max} = 2$$
$$n_{backspacing,max} = 1$$

This choice provides a high variety of sequences, such that interesting ones are not discarded.

The list of radii of pericentre of each swing-by, expressed in radii of the planet, was set to:

$$\mathbf{r}_p = \begin{bmatrix} 1.1 & 1.4 & 1.6 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

The desired escape velocity from the sphere of influence of the Earth is in the range [3.5, 4] km/s. The value 4 km/s is used also as pruning criterion. The algorithm was run three times, each time considering one of the following values of launch excess velocity:

$$\mathbf{v}_0 = \begin{bmatrix} 3.5 & 3.8 & 4 \end{bmatrix} \text{ km/s}.$$

In the same flavour as for the G to C transfer problem, the minimum value of the v_∞ at Me is the merit function. Resulting sequences, for each considered value of the launch excess velocity, are shown in Table 4.21. The most promising sequence seems to be EVVMe: in fact, not only is this sequence feasible for the whole range of launch excess velocity, but it also allows a low v_∞ at Mercury, the

lowest considering maximum two swing-bys. A third swing-by of Venus (sequence EVVVMe) would further reduce the v_∞ with respect to Mercury, however, in this section, the analysis will be limited to the EVVMe sequence only.

Table 4.21. Sequences and minimum achievable relative velocity at Mercury for each possible launch excess velocity.

Sequence					Minimum v_∞ at Me, km/s		
					$v_0 = 3.5$ km/s	$v_0 = 3.8$ km/s	$v_0 = 4$ km/s
E	V	Me			-[3]	8.31	9.96
E	V	Me	Me		-	8.31	9.96
E	**V**	**V**	**Me**		**8.25**	**7.72**	**8.75**
E	V	E	Me		11.61	13.19	10.15
E	V	Me	Me	Me	-	8.31	9.96
E	V	Me	V	Me	-	10.93	13.21
E	V	V	Me	Me	8.25	7.72	8.75
E	V	V	V	Me	7.37	7.33	7.46
E	V	V	E	Me	10.18	10.36	9.75
E	V	E	Me	Me	11.61	13.19	10.15
E	V	E	V	Me	8.43	8.77	9.18
E	V	E	E	Me	11.10	9.99	10.42

INCREMENTAL SOLUTION

The Earth-Mercury transfer does not include resonant swing-bys of the Earth, thus, the E-V leg can be modelled with a simple ballistic arc and the complete solution vector for the whole Earth-Mercury reduces to:

$$\mathbf{x} = \left[t_0, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, \gamma_2, r_{p,2}, T_3, \alpha_3 \right].$$

This choice simplifies the partial solution vector at level 1, although the number of complete revolutions before the first Venus swing-by becomes a discrete parameter of problem [96]. Solutions were computed for both for 0 and 1 complete revolutions but, in the following, only the ones with 1 revolution are presented, as they are the most interesting. Bounds for this problem are presented in Table 4.22.

The objective functions for the three levels are:

$$\begin{aligned} f_1 &= v_0 \\ f_2 &= v_0 + \Delta v_1 \\ f_3 &= g_{v_\infty,3}\left(v_\infty\right) + g_{\Delta v,3}\left(v_0 + \sum \Delta v\right) \end{aligned} \tag{4.6}$$

The parameters u_l and u_u in Eq. (4.6) are shown in Table 4.23, while $r = 50$ and $s = 0.1$ for both A_3 and C_3.

[3] A dash (-) means that the sequence is not energetically feasible for the corresponding excess velocity.

Table 4.22. Bounds for the EVVMe transfer case.

Variable	LB	UB
t_0, d, MJD2000	4500	5500
T_1, d	300	500
T_2, d	180	720
T_3, d	180	720
$r_{p,i}$, R_P	1.1	3
α_i	0.01	0.9
γ_1, rad	$-\pi/2$	$\pi/2$
γ_2, rad	$-\pi/2$	$\pi/2$

Table 4.23. Parameters *a* and *b* for the objective function f_3.

$g_{v_\infty,3}(v_\infty)$		$g_{\Delta v,3}(v_0+\Delta v)$	
u_l, km/s	u_u, km/s	u_l, km/s	u_u, km/s
$-\infty$	6	$-\infty$	4

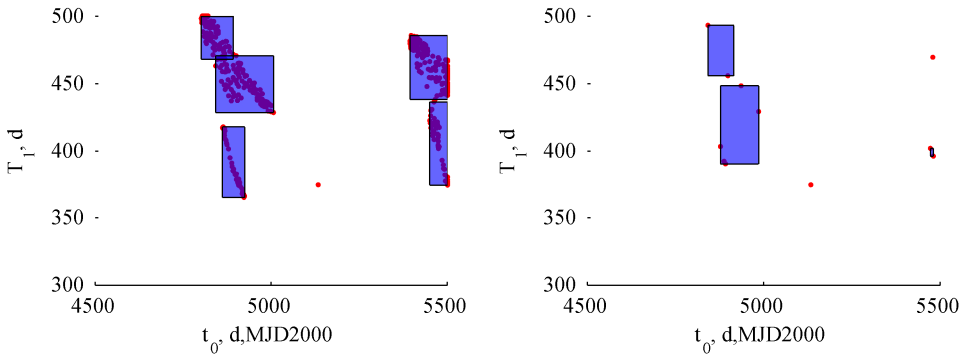
The solutions at level 1 were considered to be feasible if $v_0 < 4$ km/s (consistently with the results in Table 4.21) while at level 2, solutions were considered to be feasible if $v_0 < 4$ km/s and $\Delta v_1 < 50$ m/s. An additional pruning criterion was added at level 2 on the period of the final orbit, $P_2 < 200$ d, in order to discard all the solutions which were not reducing the period. In formulae:

$$\begin{aligned}\Phi_1 &= (v_0 < 4 \text{ km/s}) \\ \Phi_2 &= (v_0 < 4 \text{ km/s}) \wedge (\Delta v_1 < 50 \text{ m/s}) \wedge (P_2 < 200 \text{ d})\end{aligned} \tag{4.7}$$

The local optimisation was stopped when a solution with $v_0 < 4$ km/s was found for level 1, and with $v_0 < 4$ km/s, $\Delta v_1 < 50$ m/s for level 2.

Fig. 4.45 (a) and (b) represent the feasible solutions and feasible regions for level 1, after pruning level 1 and 2 respectively. Fig. 4.45 (b) shows that the back pruning removed some of the regions at level 1 when level 2 was pruned. The number of function evaluations required to obtain these results, level by level, and the corresponding computational time, are reported in Table 4.24.

If, at the end of optimisation of level 3, only solutions having $f_3 < 0.3$ (tight fulfilment of the requirements on Δv and v_∞) are selected, then only 4 options remain, which are very similar one another. One of them is presented in Fig. 4.46, and a summary of its main characteristics can be found in Table 4.25.



(a) After pruning level 1 (b) After pruning level 2

Fig. 4.45. Feasible solutions and feasible regions for variables t_0, T_1 after pruning level 1 (a) and level 2 (b).

Table 4.24. Number of function evaluation and computational time for each level of the EVVMe transfer problem.

Level	No. objective function evaluations	Time Intel	Time Sun
1	90,291	198 s	493 s
2	1,418,682	67 min	164 min
3	459,798	26 min	66 min

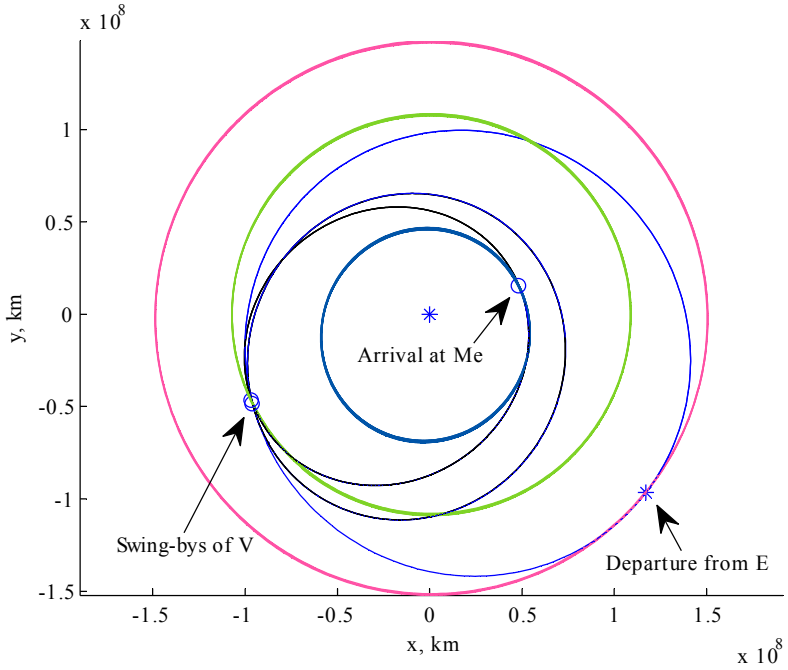


Fig. 4.46. Projection on the x-y plane of an EVVMe solution.

Table 4.25. Characteristics the EVVMe solution in Fig. 4.46.

Parameter	Value
Departure date t_0	4972.9 MJD2000 (13 Aug 2013)
Launch excess velocity v_0, km/s	3.89
T_1, d	435
T_2, d	673
Δv_1, m/s	84
T_3, d	639
Δv_2, m/s	12
Total time of flight, d	1747
Total Δv, m/s	96
v_∞ at Mercury, km/s	5.91

4.3.2 Resonant Swing-bys of Mercury

This test case is similar to the GGA2-GGA5 transfer case of the Laplace mission (Section 4.2.1). Resonant swing-bys of Mercury are used to reduce the velocity with respect to Mercury. The incoming conditions at Mercury, prior to the resonant sequence, are frozen, for sake of comparison with the ESA reference solution. Epoch and velocity used for this case are in Table 4.26. The orbit of the spacecraft before the first swing-by, and the orbit of Mercury are represented in Fig. 4.47. The reference frame is Sun centred inertial ecliptic at epoch.

The initial relative velocity with respect to Mercury (v_∞) is 5.84 km/s in magnitude and has to be brought down to at least 2.3 km/s. No requirements on final inclination or period are explicitly imposed. The total Δv and transfer time need to be minimal.

INCREMENTAL SOLUTION

The problem is tackled incrementally in an analogous way as it was explained in Section 4.2.1. Three swing-bys of Mercury are used, and the corresponding incremental problem has three levels. The solution vector has the form:

$$\mathbf{x} = \left[\gamma_1, r_{p,1}, \alpha_1, T_1, \gamma_2, r_{p,2}, T_2, \alpha_2, \gamma_3, r_{p,3}, \alpha_3, T_3\right].$$

Bounds for this problem are given in Table 4.27.

Table 4.26. Initial epoch and velocity relative to Mercury.

t_0	6717, d, MJD2000 (23 May 2018)
$\mathbf{v}_{\infty,r\theta h}$, km/s	[-2.399, 5.229, -1.002] km/s

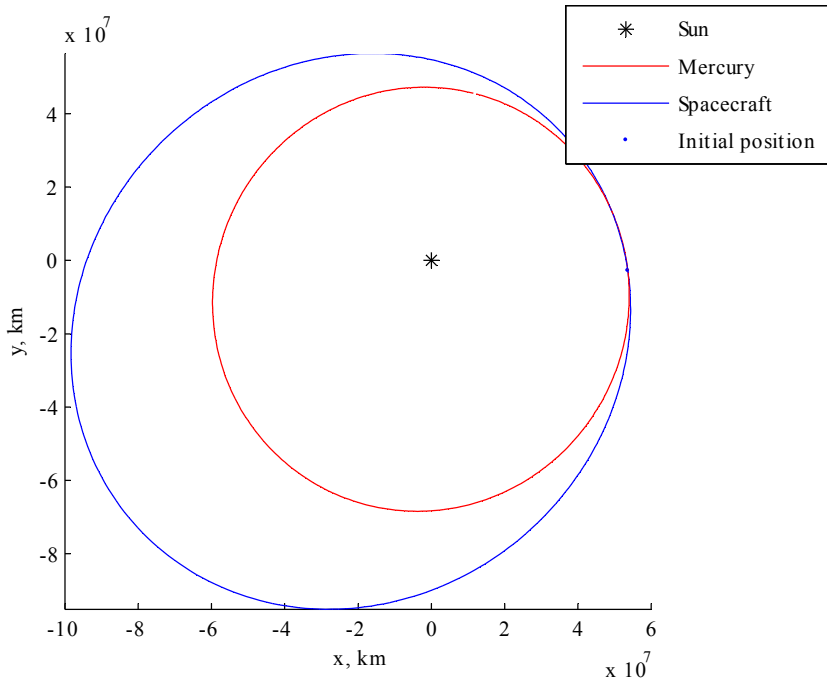


Fig. 4.47. Initial orbit of the spacecraft (before the first Me swing-by), in blue, and the orbit of Mercury (in red).

Table 4.27. Bounds for the MeMeMeMe transfer case.

Variable	LB	UB
T_1, d	45.1 ($0.5P_{Me}$)	742.5 ($8P_{Me}+20$ d)
T_2, d	90.3 (P_{Me})	561.9 ($6P_{Me}+20$ d)
T_3, d	90.3 (P_{Me})	541.9 ($6P_{Me}$)
$r_{p,i}$, R_P	1.0893 ($h_p = 200$ km)	3
α_i	0.01	0.9
γ_i, rad	$-\pi/2$	$3\pi/2$

Objective functions were chosen as following:

$$\begin{aligned} f_1 &= \Delta v_1 \\ f_2 &= \Delta v_1 + \Delta v_2 \\ f_3 &= g_{v_\infty,3}(v_\infty) + g_{\Delta v,3}(\Delta v) \end{aligned} \tag{4.8}$$

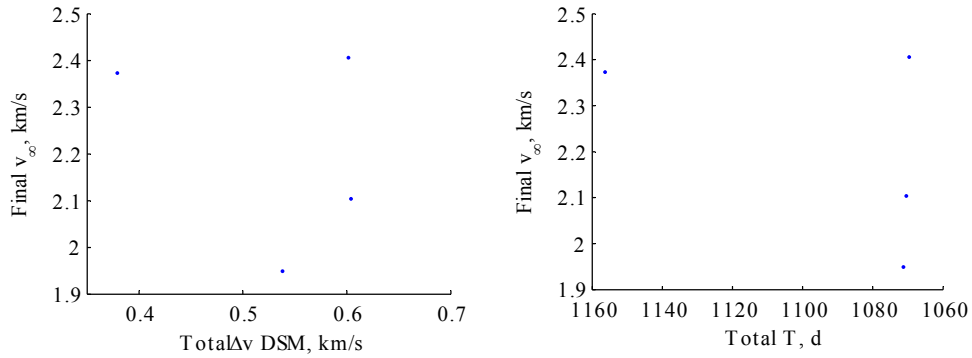
Parameters u_l and u_u for $g_{v_\infty,3}$ and $g_{\Delta v,3}$ were set as indicated in Table 4.28, and the other parameters were set to $r = 20$ and $s = 0.1$ for both functions.

Table 4.28. Parameters *a* and *b* for the objective function f_3.

$g_{v_\infty,3}(v_\infty)$		$g_{\Delta v,3}(\Delta v)$	
u_l, km/s	u_u, km/s	u_l, km/s	u_u, km/s
$-\infty$	2.4	$-\infty$	0.6

Table 4.29. Number of function evaluation and computational time for each level of the MeMeMeMe transfer problem.

Level	No. objective function evaluations	Time Intel	Time Sun
1	261,422	351 s	933 s
2	655,748	23 min	59 min
3	1,888,860	89 min	231 min



(a) Δv vs. v_∞ (b) T vs. v_∞

Fig. 4.48. Solutions to the MeMeMeMe transfer problem.

The following pruning criteria were adopted for each level. For level 1, solutions were considered feasible if $\Delta v_1 < 300$ m/s and $v_\infty < 5.83$ m/s. For level 2, solutions were feasible if $\Delta v_1, \Delta v_2 < 300$ m/s and $v_\infty < 5.6$ m/s. These pruning criteria translate into:

$$\begin{aligned} \Phi_1 &= (\Delta v_1 \le 300 \text{ m/s}) \wedge (v_\infty < 5.83 \text{ m/s}) \\ \Phi_2 &= (\Delta v_1, \Delta v_2 \le 300 \text{ m/s}) \wedge (v_\infty < 5.6 \text{ m/s}) \end{aligned} \tag{4.9}$$

In addition, each local optimisation in the Multi-Start was stopped when a solution satisfied the condition $\Delta v_1 < 300$ m/s at level 1, and the condition $\Delta v_1, \Delta v_2 < 300$ m/s at level 2.

The number of function evaluations required to obtain these solutions, level by level, and the corresponding computational time, are reported in Table 4.29. As a result of the optimisation of level 3, a total of four solutions were found, whose value of f_3 is less than 0.5. The total Δv, time of flight and final v_∞ at Mercury of these solutions are represented in Fig. 4.48.

It has to be noted that solutions in Fig. 4.48, found with the incremental approach, do not minimise the Δv, nor the final v_∞, but a combination of the two according to the objective function f_3 in Eq. (4.8). This means that potentially these solutions could be improved by minimising the total Δv, and constraining the v_∞ to a given value. Thus, for each solution, an optimisation was run constraining v_∞ to 2.3, 2.2, 2.1, 1.9, 1.8 km/s and minimising the total Δv. The re-optimised solutions having $\Delta v < 500$ m/s are shown in Fig. 4.49, in of the total Δv - v_∞ plane. The colour scale is proportional to the total time of flight of the solution. In the figure, three long transfer solutions (around 1160 days), and one short transfer solution (about 1080 days) are discernible.

A projection on the x-y plane of the trajectory of the short solution (blue circle in Fig. 4.49) is shown in Fig. 4.50, and its main characteristics are reported in Table 4.30. The figure shows also the magnitude and the position of each DSM. The resonances spacecraft: Mercury of this solution are 2:3, 3:4, 4:5 for each leg respectively. The same kind of information is provided for the long solution circled in red in Fig. 4.49. The x-y projection of the trajectory is presented in Fig. 4.51, its characteristics can be found in Table 4.31. For this solution, the resonances are 2:3, 3:4, 5:6.

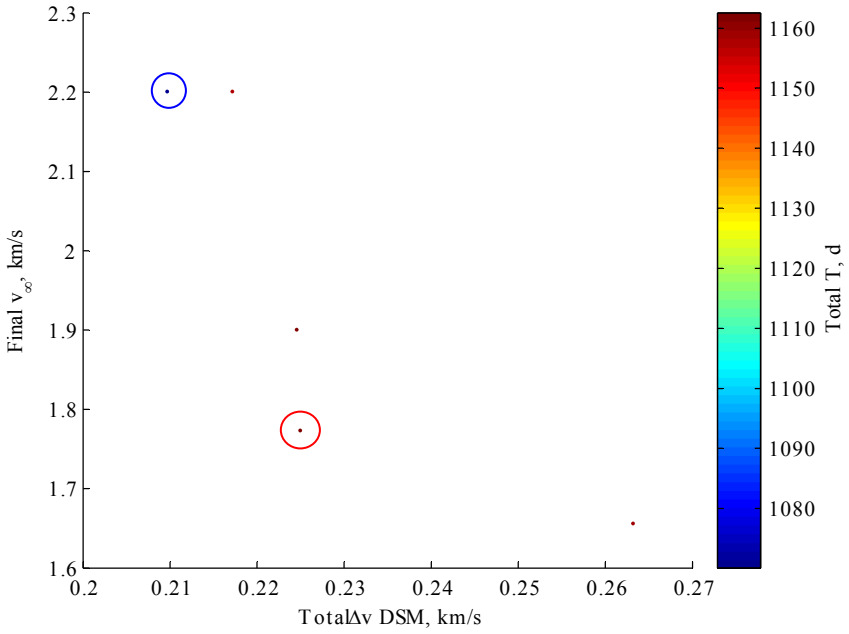


Fig. 4.49. Re-optimised solutions of the MeMeMeMe transfer problem.

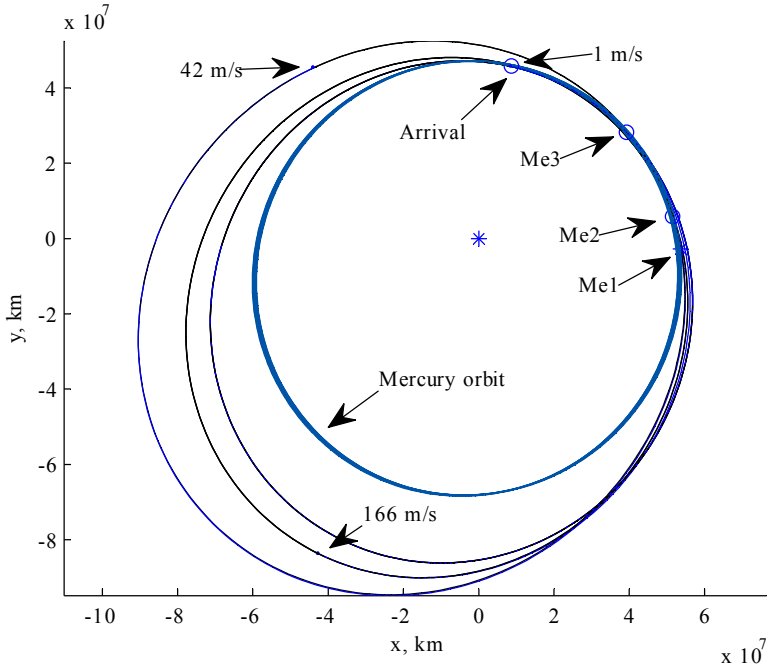


Fig. 4.50. Projection on the x-y plane of the re-optimised short solution, circled in blue in Fig. 4.49. Characteristics of this solution are presented in Table 4.30.

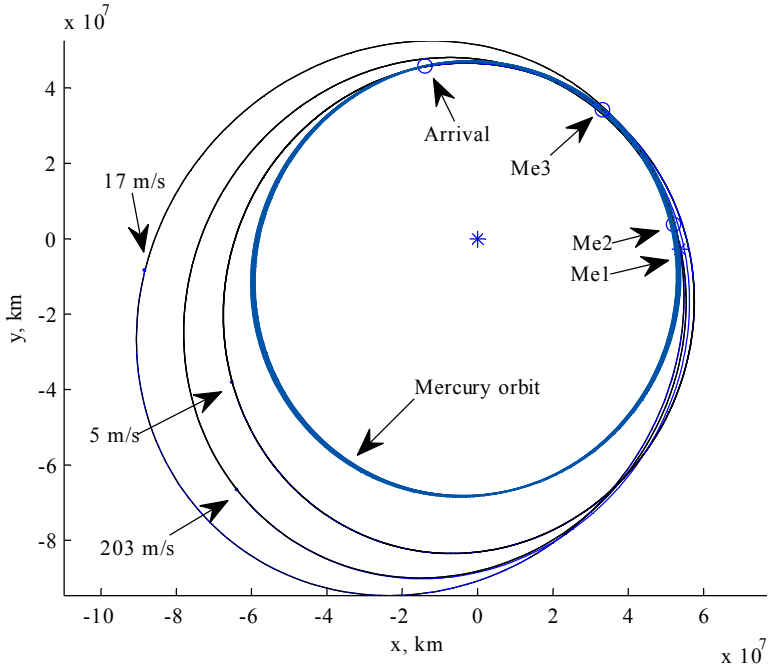


Fig. 4.51. Projection on the x-y plane of a re-optimised long solution, circled in red in Fig. 4.49. Characteristics of this solution are presented in Table 4.31.

Table 4.30. Characteristics of a re-optimised long MeMeMeMe solution, circled in blue in Fig. 4.49.

T_1, d	265.8
T_2, d	357.3
T_3, d	447
Total time of flight, d	1070
Total Δv, m/s	209
v_∞ at Me, km/s	2.2

Table 4.31. Characteristics of the re-optimised short MeMeMeMe solution, circled in red in Fig. 4.49.

T_1, d	265.3
T_2, d	359.5
T_3, d	537.7
Total time of flight, d	1162
Total Δv, m/s	225
v_∞ at Me, km/s	1.77

4.4 Summary

This chapter presented a set of four case studies. The design problems (initial conditions, target conditions) were taken from two missions that, at the time of the work, were under study at ESA: Laplace and BepiColombo. For both missions, two different segments were considered: a resonant swing-by phase, to reduce the relative velocity at the planet, and an interplanetary phase, to reach a planet with given final conditions. Differently from the test cases presented in the previous Chapter 3, the aim of this chapter was to demonstrate how the incremental pruning combined with the sequence generation can be helpful to tackle real mission design problems. Specific partial objective functions were developed, to progressively change the orbital parameters, level after level, in order to meet the objectives of the problem, at the end of the transfer.

In all the cases, the incremental approach was able to replicate the baseline solution found at ESA. In addition, it was able to identify a number of additional solutions, that could serve as possible alternatives or back-ups during the mission preliminary study.

In this chapter, the sequence generation process was also tested, and its ability to find a set of promising sequences was shown. For the Laplace interplanetary transfer case, a number of different sequences was also investigated with the incremental pruning algorithm, and an extensive analysis of the solutions was done, including a comparison with the Tisserand plane: it resulted that some solutions cannot be identified using this preliminary graphic technique.

Computational times were reported, to show that the incremental approach ran in at most a few hours on a common personal computer, hence avoiding the need of high-performance machines to obtain results quickly.

5

ACO-MGA

The chapter is organized as follows: the MGA planning problem will be briefly introduced, then the modified ACO algorithm (named ACO-MGA) will be described in detail with an analysis of its complexity; a discussion will follow comparing the proposed planning algorithm against standard ACO. Finally, two case studies will demonstrate the effectiveness of the proposed approach against Genetic Algorithms, Non-dominated Sorting Genetic Algorithm 2 and Particle Swarm Optimization.

5.1 Trajectory Model

The trajectory model is an integral part of the proposed integrated approach, subject of this chapter. The model is based on a two dimensional linked conic approximation of the trajectory: the trajectory is composed of a sequence of planar conic arcs linked together through discrete, instantaneous events. In particular, the sequence is continuous in position and piecewise continuous in velocity, i.e. each event introduces a discontinuity in the velocity of the spacecraft but not in its position. The discrete events can be: launch, deep space manoeuvre, swing-by, and brake.

Assuming that the trajectory is planar may seem very reductive, but in the solar system the inclination of the planet orbits on the ecliptic is very small (below 3 deg), with the exception of Mercury and Pluto (see Table 5.1). Although Pluto cannot be used for a swing-by, being the farthest of the bodies in the solar system, Mercury is definitely an appealing target, both for swing-bys and as a scientific destination. This is demonstrated by the NASA flying MESSENGER mission and the ESA BepiColombo mission, currently under study. A test case will show that this assumption is acceptable, and will lead to have good solutions even for a transfer to Mercury. Relative inclination of orbits of the bodies is low even in other systems of moons, like the four Galilean moons of Jupiter.

However, the transformation of the three dimensional motion of the planets into the planar model requires some attention. A possibility is to start from the real three-dimensional ephemerides given in Keplerian parameters, and then set to zero the inclination, without changing the other 5 orbital elements; another choice is to take the projection on the x-y plane of the real orbit. It is important to remark that the first option preserves the shape of the orbit in its plane (i.e., the semi-major axis and the eccentricity). This means that the period and the mean motion of the real orbit are the same as the 2D orbit's ones. This is particularly important for the implementation of this model, because it allows to read the ephemerides only once, and then propagate the orbital parameters analytically, gaining in execution speed. For this reason, it is chosen to set the inclination of the planets to zero, rather than taking the x-y projection.

The fact that the trajectory is two dimensional will be exploited in several ways, as it will be shown in the following. The major limitation of this model is that it cannot be used for missions which have the necessity to go out of the ecliptic plane [63], such as the ESA-NASA mission Ulysses, for example.

A final assumption of the present implementation is that all the orbits of both spacecraft and celestial bodies are direct, thus no retrograde orbits are allowed.

In summary, the proposed trajectory model is composed of: a launch from the departure celestial body; a series of deep space flight legs connected through gravity assist manoeuvres (modelled through a linked-conic approximation); a capture into an orbit at a target celestial body. Each one of these basic components will be explained in this chapter.

Table 5.1. Orbital inclination of the planets in the solar system on the ecliptic.

Planet	Inclination, deg
Mercury	7.0047
Venus	3.3946
Earth	0
Mars	1.8497
Jupiter	1.303
Saturn	2.4886
Uranus	0.77313
Neptune	1.7697
Pluto	17.151

5.1.1 Launch

The launch event is modelled as an instantaneous change of the velocity of the spacecraft with respect to the departure planet. The velocity change is given in terms of modulus v_0 (which depends on the capabilities of the launcher) and in-plane direction, specified through the angle φ_0, measured counter clockwise with respect to the planet's orbital velocity vector $\mathbf{v}_P$ at time of launch t_0.

According to Fig. 5.1, the initial relative velocity of the spacecraft, with respect to a reference frame centred in the planet and having the axes tangential and normal ($\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$) to its orbit, is

$$\mathbf{v}_{0,tn} = v_0 \begin{bmatrix} \cos\varphi_0 \\ \sin\varphi_0 \end{bmatrix}.$$

The vector is then transformed into the heliocentric Cartesian reference frame, getting $\mathbf{v}_{0,xy}$, and added to the velocity of the planet $\mathbf{v}_P$ to give the departure velocity:

$$\mathbf{v}_1 = \mathbf{v}_{0,xy} + \mathbf{v}_P .$$

The departure time and the direction φ_0 are free parameters of the model, while launch velocity modulus v_0 will be used to target the next planetary encounter and solve the phasing problem, as explained in the following.

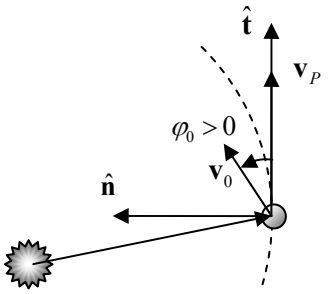


Fig. 5.1. Geometry of the launch, and convention for launch angle.

5.1.2 Swing-by

The swing-bys are unpowered, and as in the velocity formulation, they are modelled as instantaneous changes of the velocity vector of the spacecraft due solely to the gravity field of the planet. The reader can refer to Section 2.2.2 for a detailed explanation. However, in this trajectory model, the swing-by is planar and hence there is no need to define the attitude of the hyperbola plane, but only the direction of the deflection.

Given the velocity vector $\mathbf{v}^-$ before the swing-by and radius of pericentre, the anomaly of the asymptote θ_∞ can be computed with Eq. (2.6), and then the deflection angle of the asymptotic relative velocity vector, due to the planet gravity field, is:

$$\delta = \left(2\theta_\infty - \pi\right) b$$

where $b = \pm 1$ is a binary variable defining the direction of the deflection, i.e. clockwise or counter-clock wise. In fact, in the linked conic approximation the actual planetocentric hyperbola followed by the spacecraft is not defined, thus both

$2\theta_\infty - \pi$ and $\pi - 2\theta_\infty$ are acceptable deflection angles. In order to avoid introducing an additional parameter in the practical implementation on this model, we will make use of a signed radius of pericentre r_{ps} that can assume negative values, such that:

$$r_p = \left| r_{ps} \right|$$
$$b = \operatorname{sgn}\left(r_{ps} \right)$$

The outgoing relative velocity is found by rotating the incoming velocity by δ:

$$\mathbf{v}_\infty^+ = \begin{bmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{bmatrix} \mathbf{v}_\infty^-$$

And finally the absolute velocity is:

$$\mathbf{v}^+ = \mathbf{v}_\infty^+ + \mathbf{v}_P$$

As for the launch velocity magnitude, the radius of pericentre r_{ps} is tuned to meet the terminal conditions of the transfer leg following the swing-by.

5.1.3 Deep Space Flight Leg

Each deep space flight leg is made of two conic arcs linked, at a point *M*, through a single discrete event. The leg starts at a departure planet P_i and ends at an arrival planet P_{i+1}.

The event can be a deep space manoeuvre. A DSM is an instantaneous change in the heliocentric velocity vector of the spacecraft, obtained by firing an engine. In this model, we assume that the DSM is performed either at the apocentre or at the pericentre of the conic arc preceding the manoeuvre. In addition, the change in velocity is tangential to that arc. As a consequence, the DSM will raise or decrease either the pericentre or the apocentre of the orbit, without changing the line of apsides.

In this model, two different types of deep space flight legs exist, depending on whether the DSM occurs or not. If there is a DSM, then the first arc is propagated from planet P_i to point *M* for a given number of revolutions. Then, the DSM is applied, and the second arc is propagated for a number of full revolutions, until one of the intersections with the orbit of planet P_{i+1}. On the other hand, if there no DSM exists, the first arc is still propagated till a point *M*, from which, without applying any manoeuvre, the second arc starts.

FIRST ARC

Let us assume that the spacecraft is at a given planet P_1 at time t_1. Its position $\mathbf{x}_1$ coincides with that of the planet $\mathbf{x}_P$, which is known from the 2D ephemeris. The heliocentric velocity of the spacecraft $\mathbf{v}_1$, instead, depends on the preceding launch

or swing-by event. The initial state $\left[\mathbf{x}_1, \mathbf{v}_1\right]$ is converted into the 6 Keplerian elements $\mathbf{K}_1$ with inclination $i = 0$, and right ascension of the ascending node is arbitrarily set to $\Omega = 0$, since the problem is two dimensional, and retrograde orbits are not allowed.

In the case when a DSM is required (Fig. 5.2), the first step is to find the position and time of the DSM. The position can either be the pericentre or the apocentre, according to a binary variable $f_{p/a}$. The true anomaly of the DSM is straightforward:

$$f_{p/a} = \begin{cases} 0 & \rightarrow \quad \text{DSM at pericentre} \quad \rightarrow \quad \theta_{DSM} = 0 \\ 1 & \rightarrow \quad \text{DSM at apocentre} \quad \rightarrow \quad \theta_{DSM} = \pi \end{cases}$$

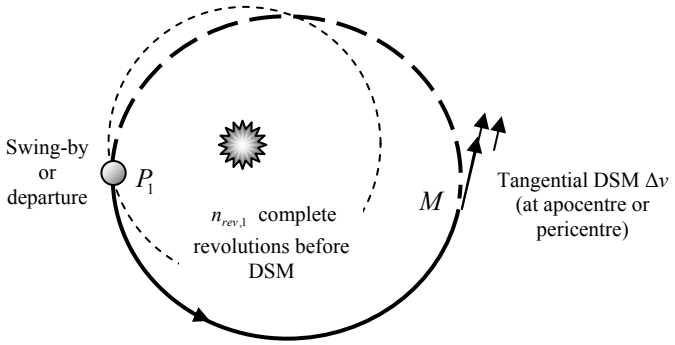


Fig. 5.2. A representation of the first arc, from the planet up to point *M*, where the DSM occurs. Possibly multiple revolutions (dashed trajectory) can be performed.

The Keplerian parameters at the point of the DSM, and before performing the manoeuvre are given by

$$\mathbf{K}_{DSM}^{-} = \left[\mathbf{K}_1(1:5), \theta_{DSM}\right].$$

The Cartesian position $\mathbf{x}_{DSM}$ of the DSM and the Cartesian velocity before performing the manoeuvre $\mathbf{v}_{DSM}^{-}$ is computed by converting $\mathbf{K}_{DSM}^{-}$. The time of the DSM is found by first computing the eccentric anomaly corresponding to the departure point θ_1:

$$E_1 = 2\arctan\sqrt{\frac{1-e}{1+e}}\tan\frac{\theta_1}{2}. \tag{5.1}$$

Then, by using the time law:

$$t_{DSM} = \sqrt{\frac{a_1^3}{\mu}}\left(2\pi n_{rev,1} + E_{DSM} - \left(E_1 - e_1\sin\left(E_1\right)\right)\right) + t_1$$

where $E_{DSM} = \theta_{DSM} + 2k\pi$, since the manoeuvre is either at pericentre or apocentre, and the integer k must be chosen such that E_{DSM} is following E_1. The quantity $n_{rev,1}$ is the number of full revolutions before the DSM.

Once t_{DSM} is determined, the DSM is applied and the velocity after performing the DSM is given by:

$$\mathbf{v}_{DSM}^{+} = \mathbf{v}_{DSM}^{-} + \frac{\mathbf{v}_{DSM}^{-}}{v_{DSM}^{-}} m_{DSM}$$

where m_{DSM} is the magnitude and direction of the DSM: if m_{DSM} is positive, the thrust is along the velocity of the spacecraft, otherwise it is against the velocity of the spacecraft. Then, the complete state of the spacecraft, after the DSM and at the beginning of the second arc is defined as:

$$\begin{aligned} t_M &= t_{DSM} \\ \mathbf{x}_M &= \mathbf{x}_{DSM}^{+} \\ \mathbf{v}_M &= \mathbf{v}_{DSM}^{+} \end{aligned}$$

If no DSM has to be performed, i.e. $m_{DSM} = 0$, the aim of the first arc is only to move the spacecraft away from planet P_1 (see Fig. 5.3). To understand the reason for this, let us assume that the leg under consideration is resonant, that is $P_1 \equiv P_2$, and since there is no DSM, we propagate directly the arc, up to the intersection with the orbit of P_2. If no full revolutions are considered, then the first intersection is at P_1, after a null time.

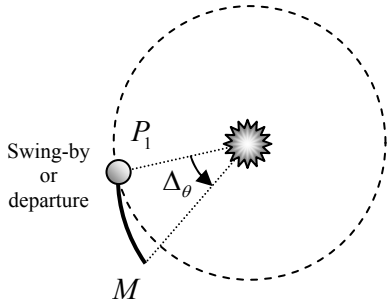


Fig. 5.3. The first arc, up to point *M*, if there is no DSM, is just an increase in the true anomaly, to move away from planet P_1.

To prevent this from happening, the conditions $[\mathbf{x}_1, \mathbf{v}_1]$ at P_1 are propagated forward, for a short amount of time. The state of the spacecraft at point *M*, $\mathbf{x}_M$, $\mathbf{v}_M$, is found starting from the Keplerian parameters:

$$\mathbf{K}_M = \mathbf{K}_1 + [0, 0, 0, 0, 0, \Delta_\theta]$$

and then converting into Cartesian coordinates. The quantity Δ_θ is a small angular displacement, larger than the machine numerical precision. For this work, $\Delta_\theta = 0.3$ rad was chosen. The time at *M* is found by solving the time law

$$t_M = \sqrt{\frac{a_1^3}{\mu}}\left(E_M - E_1 - e_1\left(\sin\left(E_M\right) - \sin\left(E_1\right)\right) + 2k\pi\right) + t_1$$

where E_1 and E_M are computed with Eq. (5.1), by using θ_1 and $\theta_M = \theta_1 + \Delta_\theta$ respectively. Again, as in the previous case, $E_M \equiv E_M + 2k\pi$, with *k* such that E_M follows E_1.

SECOND ARC

The second arc connects the spacecraft at point *M* and state $\left[\mathbf{x}_M, \mathbf{v}_M\right]$ with the proper intersection of the orbit of planet P_2 (see Fig. 5.4).

Given the orbital parameters of the spacecraft $\mathbf{K}_M$ at time t_M, and the orbital parameters of planet P_2, the task is to find the intersection between the two orbits. Two ellipses in the same plane (and here comes the assumption of planar problem) have either two (at most coincident) intersections, or no intersections. With reference to Fig. 5.5, given the polar equations of the two ellipses:

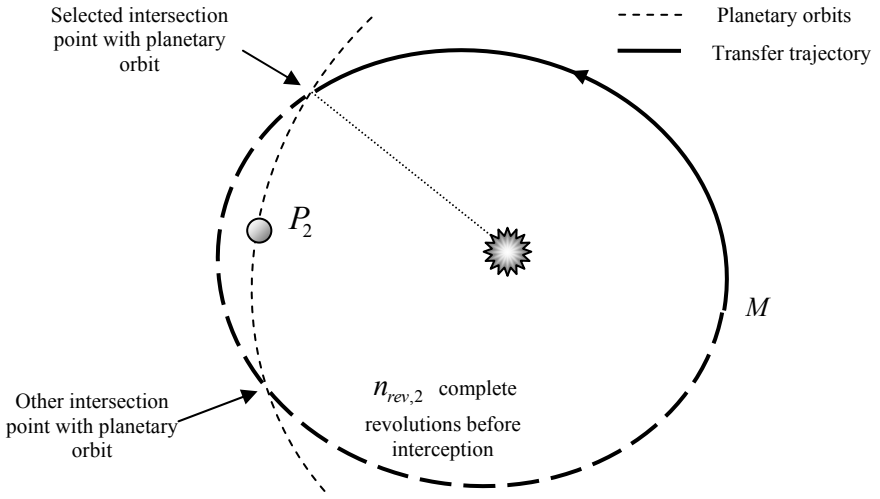


Fig. 5.4. Second arc. From point *M* (which is either after the DSM or after the first arc) to the selected orbital intersection with the planet. If $n_{rev,2} > 0$, then the full revolutions are performed (dashed trajectory) before the orbital intersection.

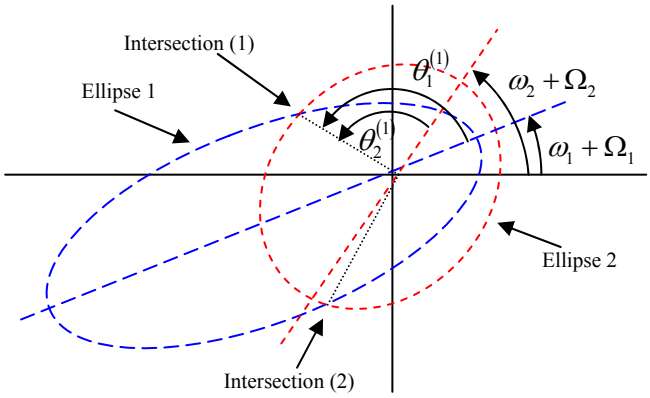


Fig. 5.5. Geometry of intersections of two coplanar ellipses (with the same focus).

$$\begin{aligned} r_1(\theta_1) &= \frac{p_1}{1+e_1\cos\theta_1} \\ r_2(\theta_2) &= \frac{p_2}{1+e_2\cos\theta_2} \end{aligned} \tag{5.2}$$

The subscripts 1 and 2 refer to the two ellipses. Let us decide that ellipse 1 is the spacecraft orbit and ellipse 2 is P_2's orbit. The intersections can be found by setting the radii and the true longitudes equal each other:

$$\begin{cases} r_1 = r_2 \\ \theta_1 + (\omega_1+\Omega_1) = \theta_2 + (\omega_2+\Omega_2) \end{cases} \tag{5.3}$$

Defining $\phi = (\omega_1+\Omega_1) - (\omega_2+\Omega_2)$ and combining Eqs. (5.2) and (5.3), after some algebra we get:

$$\underbrace{[p_1e_2\cos\phi - p_2e_1]}_{A}\cos\theta_1 \underbrace{-[p_1e_2\sin\phi]}_{B}\sin\theta_1 + \underbrace{[p_1-p_2]}_{C} = 0 \tag{5.4}$$

that is a linear equation in $\sin\theta_1, \cos\theta_1$, in which we also defined the coefficients *A*, *B* and *C*. By using the transformation $\nu = \tan\frac{\theta_1}{2}$, Eq. (5.4) becomes, after some algebra:

$$(C-B)\nu^2 + 2A\nu + (B+C) = 0$$

which has solutions:

$$\theta_1^{(1,2)} = 2\arctan\left(\frac{-A\pm\sqrt{A^2+B^2-C^2}}{C-B}\right) + 2k\pi, \qquad k\in\mathbb{Z}.$$

The equation gives the true anomaly on ellipse 1 of the two intersections. The superscripts (1) and (2) refer to the two intersections. Since we are only interested in the two intersection points, we can neglect the periodicity of the solutions, by setting $k = 0$. The true anomaly on the second orbit can be computed considering the second equation in system (5.3):

$$\theta_2^{(1,2)} = \theta_1^{(1,2)} + \phi$$

If $\Delta = A^2 + B^2 - C^2 < 0$, then there are no real solutions, which means that the spacecraft orbit does not intersect P_2's orbit, and the initial conditions of the leg, or its parameters, have to be changed.

If instead there exist two intersections, one of them is selected according to the binary variable $f_{1/2}$, which is another parameter for the leg.

$$f_{1/2} = \begin{cases} 0 & \rightarrow \quad \theta_1^{(\text{int})} = \theta_1^{(1)}, \ \theta_2^{(\text{int})} = \theta_2^{(1)} \\ 1 & \rightarrow \quad \theta_1^{(\text{int})} = \theta_1^{(2)}, \ \theta_2^{(\text{int})} = \theta_2^{(2)} \end{cases}$$

$\theta_1^{(\text{int})}, \theta_2^{(\text{int})}$ are the true anomalies on the two ellipses of the selected intersection. Now, it is possible to compute the time t_{int} of the intersection, provided that the spacecraft was at true anomaly θ_M at time t_M, by solving once again the time law, after having computed E_{int} from $\theta_1^{(\text{int})}$ using Eq. (5.1):

$$t_{int} = \sqrt{\frac{a_1^3}{\mu}} \left(2\pi n_{rev,2} + E_{int} - e_M \sin E_{int} - \left(E_M - e_M \sin E_M \right) \right) + t_M \, .$$

The integer variable $n_{rev,2}$ describes the number of full revolutions between the point *M* (possibly a DSM) and the orbital intersection (Fig. 5.4).

Finally, the Keplerian parameters at the intersection point are

$$\mathbf{K}_{int} = \left[\mathbf{K}_M \left(1:5 \right), \theta_1^{(int)} \right]$$

from which the Cartesian state $\left[\mathbf{x}_{int}, \mathbf{v}_{int} \right]$ can be computed.

5.1.4 Solution of the Phasing Problem

The fact that the spacecraft at time t_{int} intersects the orbit of planet P_2 does not imply that the planet is at the intersection point at that time, which is a requirement to perform a gravity assist manoeuvre or a planetary capture.

Therefore, for each leg, a one parameter search is started to match the terminal position of the spacecraft with that of the planet.

Depending on the event at the beginning of the leg – swing-by or launch – the following parameter λ is chosen:

$$\lambda \equiv \begin{cases} r_{ps}, & \text{if swing-by} \\ v_0, & \text{if launch} \end{cases}$$

Hence, the true anomaly of the intersection point on the planet's orbit $\hat{\theta}(\lambda)$, and the true anomaly of the planet position $\theta(\lambda)$ at the time of intersection t_{int} (see Fig. 5.6) become a function of λ. The former coincides with $\theta_2^{(\text{int})}$ computed in the previous section, while the latter is provided by the ephemeris of the planet.

The phasing problem, then, is to find one (or more) values for the parameter λ such that the planet is at the intersection point at the time of the intersection:

$$\begin{aligned} &\text{find:} && \lambda = \lambda^* \\ &\text{such that:} && \Delta\theta(\lambda^*) \equiv \theta(\lambda^*) - \hat{\theta}(\lambda^*) = 0 \end{aligned} \tag{5.5}$$

Since the true anomaly is a cyclical variable (i.e. $\theta \equiv \theta + 2k\pi$ for this purpose), care must be taken to choose the proper integer k for both θ and $\hat{\theta}$, in order to have a value of $\Delta\theta$ which allows solving the phasing problem. The procedure for computing $\Delta\theta$ is presented in Algorithm 5.1. It is worth noting that the algorithm provides correct values of $\Delta\theta$ even if the intersection point and the planet position are close to the pericentre of the orbit, but on different sides with respect to the line of apsides.

Problem (5.5) is that of finding the zero of a non-linear function. Fig. 5.7 and Fig. 5.8 represent the function $\Delta\theta(\lambda)$ for different transfer cases. Fig. 5.7 (a) and (b) are non-resonant transfers: the former is from Venus to Mercury, following a swing-by of Venus. In this case, the parameter λ is the radius of pericentre of the swing-by r_{ps}. The latter is Earth to Venus after launching from Earth, so $\lambda \equiv v_0$.

Fig. 5.8 (a) and (b), instead, refer to resonant transfers: the former is a Venus to Venus transfer starting with a swing-by; the latter is an Earth to Earth transfer, starting with launch.

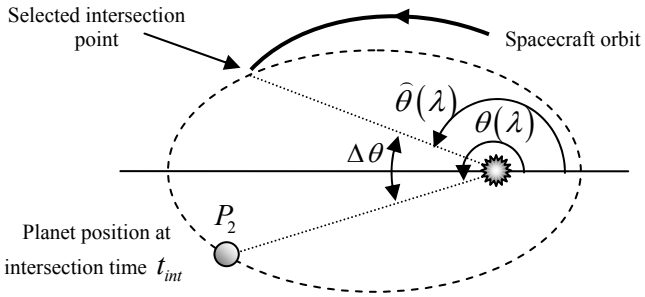


Fig. 5.6. The phasing problem consists of finding λ such that the target planet P_2 is at the orbital intersection point at the correct time. This is done by finding the zero of the difference in true anomalies $\Delta\theta$.

Algorithm 5.1. This pseudo-code illustrates the procedure for finding a meaningful value for $\Delta\theta$ regardless the cyclic nature of the two input variables.

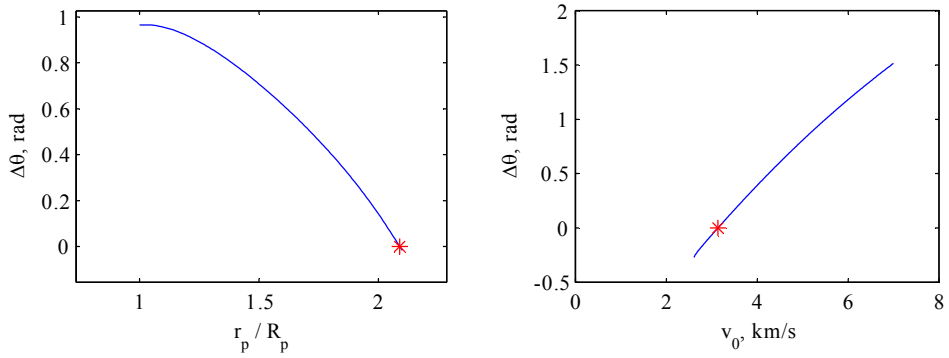
Function $\Delta\theta \leftarrow \text{ComputeDeltaTheta}\left(\theta, \hat{\theta}\right)$

1: $\theta \leftarrow \text{mod}\left(\theta, 2\pi\right)$

2: $\hat{\theta} \leftarrow \text{mod}\left(\hat{\theta}, 2\pi\right)$

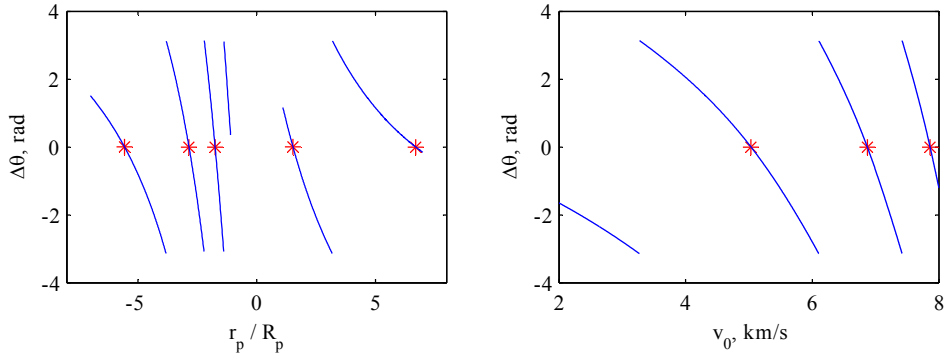
3: $\theta \leftarrow \underset{\left\{\substack{\theta \\ \theta+2\pi \\ \theta-2\pi}\right\}}{\arg\min} \left|\hat{\theta} - \theta\right|$

4: $\Delta\theta \leftarrow \hat{\theta} - \theta$



(a) Venus to Mercury (b) Earth to Venus

Fig. 5.7. $\Delta\theta(\lambda)$ for: (a) Venus to Mercury leg following a swing-by of Venus, from BepiColombo; (b) Earth to Venus leg following launch from Earth, from Cassini.



(a) Venus to Venus (b) Earth to Earth

Fig. 5.8. $\Delta\theta(\lambda)$ for: (a) Venus to Venus leg following a swing-by of Venus, from BepiColombo; (b) Earth to Earth leg following launch from Earth, from Cassini.

It is worth noting that for some values of λ, $\Delta\theta(\lambda)$ is not defined: this is the case when there is no possible orbit intersection. Examples are in Fig. 5.7 (a), for $r_{ps}/R_p > 2.1$, or Fig. 5.7 (b), for $v_0 < 2.6$ km/s. This is in fact the minimum excess velocity to reach the orbit of Venus from Earth (with $\varphi_0 = \pi$, as it was used in this case).

In the case of a leg following a swing-by, r_{ps} is also limited by the radius of the planet. This constraint translates into the condition

$$\left(\frac{r_{ps}}{R_P} < -1\right) \vee \left(\frac{r_{ps}}{R_P} > 1\right). \tag{5.6}$$

Condition (5.6) introduces the gaps in Fig. 5.7 (a) and Fig. 5.8 (a) (swing-by cases). In the cases depicted in Fig. 5.7, the function $\Delta\theta(\lambda)$ is continuous, smooth and monotonic over the range of interest of λ. Hence, the phasing problem has only one solution. This solution can be found with a simple Newton-Raphson method in one dimension.

However, when a resonant transfer is considered, as in Fig. 5.8, $\Delta\theta(\lambda)$ is discontinuous and multiple zeros exist. Each zero corresponds to a different resonance with the planet (and of course a different transfer time). The discontinuity is due to the cyclic nature of $\Delta\theta$: in fact, say λ_d is the value of λ at which $\Delta\theta$ is discontinuous, then $\lim_{\lambda\to\lambda_d^+} \Delta\theta = -\pi$, and $\lim_{\lambda\to\lambda_d^-} \Delta\theta = +\pi$, i.e. the planet and the spacecraft are on the opposite sides of the planet's orbit.

The fact that none, one or more solutions to the phasing problem (for each leg) may exist implies that a dense enough set of starting points has to be provided (both for the case $\lambda \equiv r_{ps}$ and $\lambda \equiv v_0$), and the zero-finder algorithm shall be run from each one of the starting points, to make sure that all the zeros are found; the zero finder shall not converge on the discontinuity (or at least discard the result if that happens), which would be the behaviour, for example, of the bisection method if the two starting points are chosen across the discontinuity; different solutions lead to different final conditions, and so they affect the subsequent part of the trajectory.

Since there is no easy way, at a given leg, to prefer one value of λ^* rather than another, having multiple solutions means that all of them shall be considered for the subsequent legs.

The search for the zeros of both $\Delta\theta(v_0)$ and $\Delta\theta(r_{ps})$ is performed by means of a Brent method [119]. This method resulted to be fast and robust, since it uses a Newton based method for quick convergence, but it is able to switch to a bisection-like method when needed. A set of starting points needs to be provided as a parameter to initialise the Brent method.

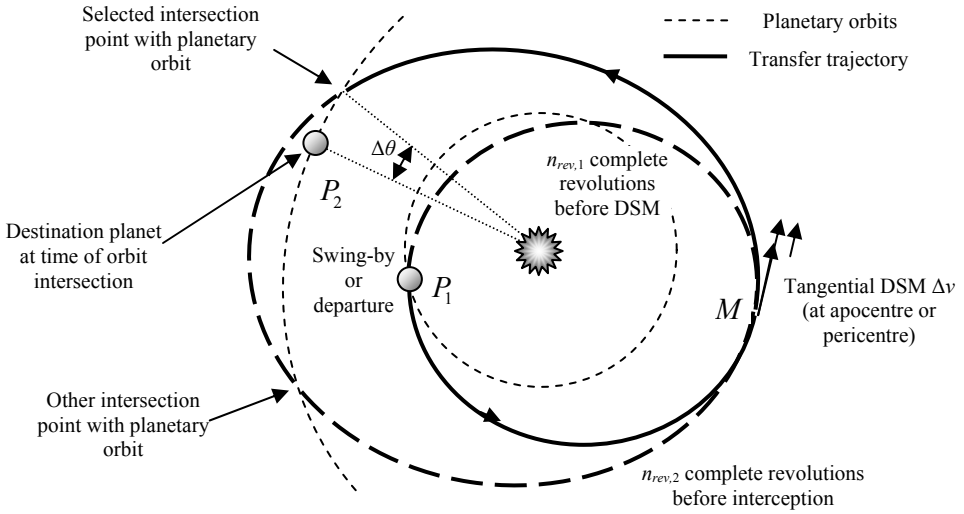


Fig. 5.9. Representation of a complete leg, with a DSM and possibly multiple revolutions. The phasing problem with P_2 is not solved.

Figure 5.9 illustrates a complete leg, including a DSM: starting from planet P_1 with either a swing-by or launch, the first arc is propagated for a number of full revolutions until the point *M* (the apocentre of the orbit in the picture). The DSM raises the pericentre and the second arc, after some full revolutions, intersects the orbit of planet P_2. The figure also shows that the phasing problem is not solved, as P_2 at the time of intersection is not at the intersection point.

5.1.5 Complete Trajectory

The trajectory model requires the ephemerides of the planets, the gravitational constant and the radius of the main attracting bodies.

The sequence of planetary encounters is characterised by the vector of integer values $\left[P_1, P_2, \ldots P_{n_P}\right]$. The interplanetary transfer starts with a launch at P_1, followed by a interplanetary leg from P_1 to P_2. Then a number of swing-bys and interplanetary legs follow, starting from P_2 until the target planet P_{n_p} is reached. It follows that a sequence with n_P planets has $n_{legs} = n_P - 1$ legs, and $n_{legs} - 1$ swing-bys.

The departure time t_0 and departure angle φ_0 are free design variables, together with five variables for each leg: 1 real (m_{DSM}), 2 integers ($n_{rev,1}, n_{rev,2}$), and 2 binaries ($f_{p/a}, f_{1/2}$). Since these five variables fully characterise the interplanetary leg, each set of values of these variable represents one *type of transfer*. In Table 5.2, the design variables for the entire trajectory are listed. The set of these variables define a *solution* to the transfer problem.

Table 5.2. Description of the free design variables defining a solution according to the proposed 2D model.

Description	Variables
Planetary sequence, with P_{n_p} planets and $n_{legs}-1$ swing-bys	$\left[P_1, P_2, \ldots P_{n_P}\right]$
Departure time	t_0
Departure angle	φ_0
Set of 5 of variables to characterise each leg $i = 1, \ldots, n_{legs}$: 1 real, 2 integers, 2 binaries	$\left[m_{DSM}, n_{rev,1}, n_{rev,2}, f_{p/a}, f_{1/2}\right]_i$

Algorithm 5.2. This pseudo-code generates a list *L* containing the arrival conditions for all the feasible trajectories of the transfer problem.

```
P_1 → P_2: find all possible v_0^* | Δθ(v_0^*) = 0
For each v_0^* find the final conditions of the leg
    Add all the possible final conditions to list L
End For
For each leg i = 2,...,n_leg
    L_temp ← ∅
    For each element in list L
        P_i → P_{i+1}: find all possible r_ps^* | Δθ(r_ps^*) = 0
        For each r_ps^*
            Find the final conditions at planet P_{i+1} and add to list L_temp
        End For
    End For
    If L_temp = ∅, Then All trajectories infeasible at leg i, Terminate
    L ← L_temp
End For
```

In addition to the design variables, two vectors containing the starting guess values for v_0 and r_p need to be provided.

It was pointed out that at launch or at each swing-by, multiple zeros may arise in the phasing problem. Each zero generates a different trajectory for the leg, thus creating a tree-like structure for the whole transfer problem, in which every branch is a different trajectory. Algorithm 5.2 illustrates the procedure to keep track of the tree of possible trajectories, as long as legs are added to the solution. At the end of the algorithm, the list L contains all the possible conditions of arrival at the last planet in the sequence.

Since the time of flight is not explicitly bounded within this model, an upper bound of the time of flight for the entire trajectory or for some legs is introduced:

all the solutions that exceed the total or partial time of flight constraint are discarded from the list.

Note the termination condition at line 13: if at a given leg *i*, there are no solutions which satisfy the phasing problem, then the trajectory is unfeasible at leg *i*, and the algorithm terminates. The information of unfeasibility at a given leg will be used to fill in a taboo list of broken and impracticable solutions.

For each of the trajectories found, the model computes:

- The sum of all the DSMs, or total Δv and the launch excess velocity, v_0, which is the result of the phasing problem for the first leg;
- The relative velocity at the last planet, v_∞. This value is usually important for assessing the optimality of a trajectory, as a low v_∞ implies that a small manoeuvre is needed for the spacecraft to be captured by the target planet;
- The total time of flight of the trajectory. The total time of flight is important when assessing the trajectory, as long missions may not be feasible due to excessive cost of the operations.

The whole model was implemented in ANSI C and compiled as a MEX-file for interfacing with MATLAB®.

5.2 The ACO-MGA Algorithm

The construction of an optimal MGA trajectory is translated into an optimal planning problem with finite state space. An optimisation procedure, based on the ant colony optimisation paradigm, was developed to explore the space of possible plans. A plan is fully defined by assigning a value to the parameters in Table 5.2 for all the possible legs connecting the departure body to the target one. A complete plan will be called a *solution* of the optimal planning problem in the following. A partial or incomplete plan is the set of parameters sufficient to describe a solution up to a given leg, and will be referred to as partial solution. A plan does not define a single scheduled sequence of events, as the scheduling of the events is solved internally in the model by solving the phasing problem presented in the previous sections. Therefore, each plan, or solution, corresponds to a number of scheduled sequences of events or *trajectories*. Each trajectory is characterised by a different combination of v_0^* and r_p^* for each leg.

The optimisation algorithm, called ACO-MGA in the following, operates a search in the finite space of possible values for the design variables in Table 5.2, including the planetary sequence **s**, for a fixed number of legs. A complete description of the algorithm ACO-MGA, that was implemented in MATLAB®, follows.

5.2.1 Solution Coding

In ACO-MGA, a solution is coded through a string of discrete values assigned to the parameters in Table 5.2. However, the set of parameters is inhomogeneous, as it

is made of real, integer and binary variables. In particular, t_0, φ_0 and $m_{DSM}\big|_i$ are real continuous variables and need to be properly discretised. In the present implementation of ACO-MGA, the values of the departure date t_0 and the departure angle φ_0, are assumed to be pre-assigned and therefore the two parameters are removed from the list of the variables. The rationale behind this choice is that, although the launch date has a great impact on the resulting trajectory, if an algorithm exists that is able to efficiently generate a complete plan for a given launch date, a systematic search can be performed along the launch window, with a given time step.

The angle φ_0 on the other hand can very often be estimated depending on the mission: usually a tangential launch is used for non-resonant legs, in order to maximise the change in semimajor axis. The launch will be in the same direction of the planet heliocentric velocity ($\varphi_0 = 0$) if the second planet in the sequence is outwards; vice versa, the launch will be in the opposite direction ($\varphi_0 = \pi$) if the second planet is inwards [74]. For resonant legs, instead, very often $\varphi_0 = \pm\pi/2$ is chosen, as this value allows maximising the radial component of the relative velocity vector, at the following swing-by [102]. Furthermore, it is assumed that the departure planet P_1 is pre-assigned, as it is in the great majority of the applications.

Using the additional assumptions on t_0, φ_0 and P_1, each solution can be coded using a vector $\mathbf{x}$ of positive integers. The vector has $2n_{legs}$ entries. Each pair of consecutive entries encodes all the parameters necessary to characterise one leg of the solution (Fig. 5.10). The first element of the pair is encoding the identification number of the target planet for the corresponding leg according to the following procedure: an ordered list $\mathbf{q}_{P,i}$ containing all the planets available as a target for each leg *i* is predefined (and fixed); then, said $k = x_{2(i-1)+1}$, the target planet is the n^{th} entry in the list $\mathbf{q}_{P,i}$, i.e. $q_{P,(i,k)}$.

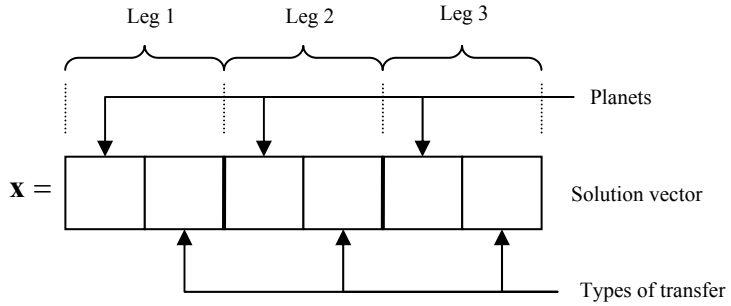


Fig. 5.10. Vector for coding a three-leg solution.

The second element of the pair is the row index of a matrix containing all possible combinations of values of the five discrete variables

$m_{DSM,i}, n_{rev,1,i}, n_{rev,2,i}, f_{p/a,i}, f_{1/2,i}$ and thus it defines the type of transfer. For each leg, let us assume that the value of each of the five parameters belongs to finite pre-defined ordered sets:

$$m_{DSM,i} \in \mathbf{q}_{1,i}; n_{rev,1,i} \in \mathbf{q}_{2,i}; n_{rev,2,i} \in \mathbf{q}_{3,i}; f_{p/a,i} \in \mathbf{q}_{4,i}; f_{1/2,i} \in \mathbf{q}_{5,i} \tag{5.7}$$

and define the vector $\mathbf{n}_{par,i}$ whose components are the cardinalities of each of the sets in (5.7):

$$\mathbf{n}_{par,i} = \left[\left|\mathbf{q}_{1,i}\right| \quad \left|\mathbf{q}_{2,i}\right| \quad \left|\mathbf{q}_{3,i}\right| \quad \left|\mathbf{q}_{4,i}\right| \quad \left|\mathbf{q}_{5,i}\right| \right]$$

where $\left|\square\right|$ indicates the cardinality. From this vector, it is possible to build the Cartesian product matrix:

$$\mathbf{G}_i = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left|\mathbf{q}_{1,i}\right| & \left|\mathbf{q}_{2,i}\right| & \left|\mathbf{q}_{3,i}\right| & 2 & 2 \end{bmatrix} \tag{5.8}$$

For the matrix $\mathbf{G}_i$ in Eq. (5.8), it was assumed that $\left|\mathbf{q}_{4,i}\right| = \left|\mathbf{q}_{5,i}\right| = 2$, since $f_{p/a,i}$ and $f_{1/2,i}$ are binary variables. If the position of the DSM on the orbit (pericentre or apocentre) or the position of the planetary intersection can be determined a priori, then $\left|\mathbf{q}_{4,i}\right|, \left|\mathbf{q}_{5,i}\right| \le 2$. Each row of $\mathbf{G}_i$ is a vector representing a different type of transfer. In general, the matrix has $\prod_j n_{par}(j)$ rows, which is also the number of different transfers for a given leg. The parameters for the j^{th} type of transfer can be obtained as follows:

$$\begin{aligned} m_{DSM,i} &= \mathbf{q}_{1,(i,k_1)} \\ n_{rev,1,i} &= \mathbf{q}_{2,(i,k_2)} \\ n_{rev,2,i} &= \mathbf{q}_{3,(i,k_3)} \\ f_{p/a,i} &= \mathbf{q}_{4,(i,k_4)} \\ f_{1/2,i} &= \mathbf{q}_{5,(i,k_5)} \end{aligned}$$

where $k_l = G_{i,(j,l)}$. Note that in ACO-MGA this process is repeated for each leg *i* of the transfer.

5.2.2 The Taboo and Feasible Lists

A given leg i depends on the preceding legs from 1 to $i-1$; thus, the leg from planet P_i to planet P_{i+1} can be feasible or unfeasible, for the same set of transfer parameters, depending on all the preceding legs. For this reason, when an infeasible leg is generated, it is necessary to store the path that led to that infeasible leg. Thus, all the variables characterising the partial solution preceding the unfeasible one are stored in a taboo list. In particular, if the problem involves n_{legs} legs, then the same number of taboo lists are used. The taboo list of leg i contains all the partial solutions which are found to be unfeasible due to no feasible trajectory at leg i (but feasible for legs $1,...,i-1$). Thus, the taboo list for each leg is stored in a matrix, which has an arbitrary number of rows and a number of columns suitable to store a partial solution up to the corresponding leg, i.e. $2i$.

The number of elements in the taboo lists can be limited, to limit the memory requirements and the search time. Once one of the taboo lists is full, the optimiser can either stop or simply start replacing the older elements.

Dual to the list of taboo partial solutions, the feasible list stores all the solutions, which are completely feasible, i.e. reach the destination planet. This is once more a matrix with an arbitrary number of rows and $2n_{legs}$ columns.

Since each solution contained in the feasible list is complete, then it is possible to associate an objective value to each one of them because the value of the launch excess velocity v_0, all the DSMs, the arrival relative velocity v_∞, and the time of flight T are available. A scalar value can be computed from these quantities identifying the cost of the trajectories. In the following test cases, for example, we will use, as objective value, the v_∞ and a combination of v_∞ and T.

Note that, since, in general, there is more than one trajectory for a given solution (i.e. for a given set of free design variables), the objective value of a solution is given by the best trajectory value.

As for the taboo list, the feasible list length can also be limited for memory saving. In this case, when the list is full, the optimisation can either stop or simply replace the feasible solutions with the worst (highest) objective value.

5.2.3 Search Engine

The search space is organised as an acyclic oriented tree. Each branch of the tree represents a leg of the problem, while each node (or leave) represents a different destination planet and type of transfer. A population of virtual ants are dispatched to explore the tree, searching for an optimal solution.

The search runs for a given number of iterations $n_{iter,max}$, or until a maximum number of objective function evaluations $n_{eval,max}$ has been reached. An evaluation is a call to the model, in order to compute the objective value associated to a given solution.

Algorithm 5.3 illustrates the main iteration loop. Each iteration consists of two steps: first, a solution generation step (lines 2 to 6), and then a solution evaluation

step (line 7). In the former step, the ants incrementally compose a set of solution vectors, while the latter invokes the trajectory model to assess the feasibility and the objective value of each generated solution. When the main loop of the search stops, the feasible list contains all the solutions, which were found feasible, with their corresponding objective value. The solutions are then sorted according to their objective value.

Algorithm 5.3. Main ACO-MGA search engine.

1: **While** $n_{iter} < n_{iter,max} \wedge n_{eval} < n_{eval,max}$, **Do**
2: **For** each ant $k = 1 \ldots m$
3: $\mathbf{x}_{temp} \leftarrow$ Generate planetary sequence (Algorithm 5.4)
4: $\mathbf{x}_{temp} \leftarrow$ Generate types of transfers (Algorithm 5.6)
5: If $\mathbf{x}_{temp}$ is not discarded, $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{x}_{temp}$
6: **End For**
7: Evaluate solutions $(\mathbf{S})$ (Algorithm 5.7)
8: Update feasible list and taboo lists
9: Update n_{iter}, n_{eval}
10: **End Do**
11: Sort feasible list according to $\boldsymbol{y}$.

SOLUTION GENERATION

The tree is simultaneously explored, from root to leaves, by m ants. At each iteration, each one of the m ants explores the tree independently of the others, but taking into account the information collected by all the ants at the previous iterations, through the feasible list and the taboo lists. As an ant moves along a branch, it progressively composes a complete solution. At first, each ant assigns a value to the odd entries of the solution vector, i.e. composes the sequence of planetary encounters, then it assigns a value to the even entries of the solution vector, i.e. the parameters defining the type of transfer for each legs.

Planetary sequence generation

The process is described in Algorithm 5.4. Each ant composes a solution adding one planet at the time. As the departure planet is given, the ant has only to choose the destination planet for each leg. The choice is made probabilistically by picking from the list $\mathbf{q}_{P,i}$ of possible celestial bodies for the i^{th} leg. The selection depends on the pheromone distribution vector $\boldsymbol{\tau}_i$ (one for every leg) which contains the pheromone level associated to each body in $\mathbf{q}_{P,i}$. Note that we use the same notion of pheromone as in standard ACO [87], however there are some differences. Here, the pheromone level of each possible choice at each leg depends on the previous legs,

and therefore it is computed at every step. Furthermore, due to the different pheromone update rule, here the amount of pheromone is not upper limited to 1.

Algorithm 5.4. The code generates the planetary sequence of the temporary solution probabilistically.

```
For each leg i = 1...N_l
    τ_{P,i} ← [1  1  …  1]
    For each target body j available at leg i
        x_temp(1 + 2(i − 1)) ← j
        For each solution l in the feasible list that matches x_temp
            τ_{P,i,j} ← τ_{P,i,j} + (1/y_l) w_planet
        End For
    End For
    x_temp(1 + 2(i − 1)) ← SelectProbabilityDistribution(τ_i)
End For
```

At every generation, each ant builds a temporary solution vector $\mathbf{x}_{temp}$ by incrementally filling in the odd components. Every time an ant is at leg i, the pheromone distribution vector is reset to $\boldsymbol{\tau}_i = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$. As it will be explained, this is equivalent to state that all the planets have equal probability to be chosen. The ant sweeps the entire list $\mathbf{q}_{P,i}$ substituting the identification number of each element in $\mathbf{q}_{P,i}$ into the i^{th} odd component of the partial solution vector $\mathbf{x}_{temp}$. Then, the feasible list is searched for all the solutions which have a (partial) planetary sequence which matches the one in $\mathbf{x}_{temp}$. Say that the j^{th} element of $\mathbf{q}_{P,i}$ is added to $\mathbf{x}_{temp}$, and the partial sequence in $\mathbf{x}_{temp}$ matches the partial sequence of the l^{th} solution in the feasible lists, then the pheromone level $\tau_{P,i,j}$ associated to the j^{th} element of $\mathbf{q}_{P,i}$ is increased as follows:

$$\tau_{P,i,j} \leftarrow \tau_{P,i,j} + \frac{1}{y_l} w_{planet} \tag{5.9}$$

The amount of pheromone deposited which is added depends on the objective value y_l of the matching solution in the feasible list, and on the weight w_{planet}. Once the pheromone update has been done for all the possible choices, the probability of selecting one of them is given by $Pr_{P,i,j} = \tau_{P,i,j} \Big/ \sum_j \tau_{P,i,j}$, and a random selection is performed according to this discrete probability distribution. Thus, the probability of choosing the j^{th} planet increases according to how many

times it generates a promising sequence (leading to a feasible solution), to the value of the feasible solution itself, and to the parameter w_{planet}.

This mechanism is analogous to the pheromone deposition of standard ACO and aims at driving the search of the ants toward planetary sequences, which previously led to good solutions. In fact, those planets which generate (partial) sequences that appear either frequently in the feasible list, or rarely, but with low objective function are selected with higher probability. On the other hand, the probability of selecting other planets from $\mathbf{q}_{P,i}$ remains positive, such that one or more ants can probabilistically choose a planet that generates an undiscovered sequence. Note that, if the feasible list is empty, then all the planets have the same probability to be selected.

The parameter w_{planet} controls the learning rate of the ants. A low value of w_{planet} will make the term w_{planet}/y_l small, and thus the probability distribution will not change much, even if the solution appears repeatedly in the feasible list, or with low values of y. Thus, a relatively low value of w_{planet} will favour a global exploration of the search space, while a high value of w_{planet} will greatly increase the probability of choosing a planet which led to a feasible sequence. If the value of w_{planet} is high enough with respect to a reference objective value, then the ant will preferably choose a feasible sequence, rather than trying a new one, which has not proven to be feasible. For these reasons, we can say that low values of w_{planet} will favour local exploration of planetary sequences.

Algorithm 5.5 shows how a choice is made among several, given the (non-normalised) probability distribution vector among all the possibilities.

The procedure iterates for all the legs of the problem. At the end, all the odd entries of the temporary solution $\mathbf{x}_{temp}$ contain a target planet and the planetary sequence is completed. The next step is to find the type of transfers for each leg, thus filling the even entries of $\mathbf{x}_{temp}$ and complete the solution.

Algorithm 5.5. This function chooses a random value from a discrete distribution function [87].

Function $j \leftarrow \text{SelectProbabilityDistribution}(\boldsymbol{\tau})$

1: $s \leftarrow \sum_j \tau_j$
2: $r \leftarrow \text{rand} \cdot s$
3: $j \leftarrow 1$
4: $p \leftarrow \tau_1$
5: **While** $p < r$
6: $\quad j \leftarrow j + 1$
7: $\quad p \leftarrow p + \tau_j$
8: **End While**

Type of transfer generation

Once a ant has composed a temporary solution $\mathbf{x}_{temp}$, containing the planetary sequence, it proceeds assigning the values associated to the type of transfer for each leg. Similarly to the planet sequence generation, for each leg, all the available types of transfer are assigned, one at the time, to the temporary solution $\mathbf{x}_{temp}$. A vector $\mathbf{x}_{temp}$ for which a value is assigned to both the odd and even component up to leg *i* represents a partial solution. For each new partial solution, at first the taboo list is checked. If the partial solution appears in the taboo list, then it means that this solution will be unfeasible, regardless of the way the solution is completed. The pheromone level of the type of transfer associated to that sequence is set to zero, to avoid the future selection of this type of transfer. If the partial solution does not appear in the taboo list, the feasible list is searched for feasible solutions which match the partial solution $\mathbf{x}_{temp}$. This time, the solution has to match up to the considered leg, both for the planetary sequence and for the types of transfer.

For every match found, the pheromone level for that type of transfer is modified as follows:

$$\tau_{t,i,j} \leftarrow \tau_{t,i,j} + \frac{1}{y_l} w_{type}$$

The weighing w_{type} is introduced, with analogous meaning to w_{planet}, to regulate the learning rate of the type of transfer: in fact, the higher the coefficient, the higher the chances that solutions similar to the feasible ones are generated. Conversely, a low value of w_{type} will favour the selection of sequences with a different type of transfer, thus increasing the random exploration of the whole solution space.

If, at a given leg *i*, all possible transfer types correspond to partial solutions which are in the taboo list, the vector of pheromone distribution $\boldsymbol{\tau}_{T,i}$ will be full of zeros. As a consequence, the solution $\mathbf{x}_{temp}$ (which can be partial or complete) is discarded, and the ant can stop its exploration of that branch of the tree.

At the end of the solution generation step, the solution $\mathbf{x}_{temp}$ is either discarded or completed. Once all the ants completed their exploration, the result is a number of solutions (less than or equal to the number of ants *m*) to be evaluated. The procedure is summarised in Algorithm 5.6.

SOLUTION EVALUATION

Once a set of solutions **S** has been generated by the ants, each solution has to be evaluated to assess its feasibility and its objective value. This is done by calling the trajectory model. Algorithm 5.7 illustrates the procedure for evaluating the solutions and storing their feasibility and objective values.

Algorithm 5.6. The code generates the types of transfer of the temporary solution probabilistically.

For each leg $i = 1 \ldots n_{legs}$
  $\boldsymbol{\tau}_{T,i} \leftarrow [1 \quad 1 \quad \ldots \quad 1]$
  For each type of transfer *j* available for leg *i*
    $x_{temp}(2 + 2(i-1)) \leftarrow j$
    If $\mathbf{x}_{temp}$ is in taboo list of leg *i* **Then**
      $\tau_{T,i,j} \leftarrow 0$
    Else
      For each solution *l* in the feasible list that matches with $\mathbf{x}_{temp}$
        $\tau_{T,i,j} \leftarrow \tau_{T,i,j} + \frac{1}{y_l} w_{type}$
      End For
    End If
  End For
  If $\sum_j \tau_{T,i,j} = 0$ **Then**
    Discard this solution, **Terminate**
  Else
    $\mathbf{x}_{temp}(2 + 2(i-1)) \leftarrow \text{SelectProbabilityDistribution}(\boldsymbol{\tau}_i)$
  End If
End For

Algorithm 5.7. Solution evaluation.

Remove duplicate solutions from the set $\mathbf{S}$
For each solution $\mathbf{S}(i) \in \mathbf{S}$
  $[y, l_{unf}] \leftarrow \text{EvaluateSolution}(\mathbf{S}(i), \mathbf{p})$
  If $l_u > 0$
    Put $\mathbf{S}(i)$ in taboo list of leg l_{unf}
  Else
    Put $\mathbf{S}(i)$ in feasible list, with objective value *y*
  End If
End For

Before evaluating the solutions in the set, duplicates are removed: in fact, since the ants explore simultaneously and without directly sharing any information among each other, there is the possibility of having ants generating the same solution during the same iteration.

At this point, solutions in **S** are evaluated one by one, by means of the model presented before. The trajectory model can be seen as a function which takes a

solution vector **x** as an input, and gives as an output either an objective value y (if the solution is feasible) or the leg at which the solution becomes unfeasible l_{unf}:

$$\left[y, l_{unf} \right] \leftarrow \text{EvaluateSolution}\left(\mathbf{x}, \mathbf{p} \right)$$

where $l_{unf} = 0$ if the solution is feasible and **p** is the constant vector containing the values of t_0 and α_0. If the solution is feasible, then it is stored in the feasible list, otherwise it is stored in the $l_{unf}{}^{\text{th}}$ taboo list.

5.2.4 Comparison with Standard ACO

The way in which the ants generate the solutions (or tours, to use ACO nomenclature) is similar to what happens in the travelling salesman problem (TSP) through ACO. In its basic form, ACO is a meta-heuristic to tackle hard combinatorial problems. The inspiring source of ACO is the foraging behaviour of real ants [89].

Unlike in genetic algorithms, where a population of agents evolve simultaneously, interact with the others (e.g. crossover) and are recombined to generate the offspring, in ACO each artificial ant explores the space independently of the others. The ant generates the tour by adding nodes (or cities) one at a time. Each node is chosen probabilistically among a set of available nodes: for the TSP, the available nodes are the cities which have not been visited in the current tour. The probabilistic model of ACO is called the pheromone model. The pheromone value update makes the search process that is performed by ACO algorithms adaptive, in the sense that the accumulated search experience is used in order to direct the future search process.

For the MGA problem, nodes are all the possible pairs of bodies and types of transfers for each leg. For both standard ACO and ACO-MGA, the probability is distributed among all the possible choices, and then a selection is made, according to the probability distribution. In the case of ACO, the probability of each city is computed by taking into account the heuristic information and the pheromone relative to the edge connecting the current city to the next city. In the case of the MGA problem, instead, the generation of the solution is done in two steps: the first to determine the planetary sequence and the second to determine the type of transfer. Both the steps use the same approach: the probability is computed by taking into account the objective value of all the feasible solutions which share the same partial solution. In addition, taboo lists are checked to avoid solutions known to be unfeasible. Taboo lists have no equivalent in classical ACO, as for the TSP all the solutions are feasible. Furthermore, the ants are allowed to visit the same tour more than once, as this will reinforce the amount of pheromone along the whole tour.

The evaluation step can be seen as the analogous of the “pheromone update” in ACO, with a difference. In the MGA problem, as opposed to the TSP tackled with ACO, the pheromone cannot be assigned to single legs of the trajectory: this is due to the fact that each leg (identified by its couple of integers) has no intrinsic value in

the trajectory, if disconnected from the previous legs. In other words, the only two parameters (planet and type of transfer) are not sufficient to fully characterise one leg: the actual value of the leg depends also on its initial conditions, which are in turn determined by the parameters of the previous legs.

To explain this idea, we make reference to Fig. 5.11 and Fig. 5.12: the former shows a typical instance of the TSP. In this problem, the distance between each pair of cities is fixed, and it can be computed given the instance of the problem itself: the relative distances of n cities can be stored in a $n \times n$ matrix [87]. Moreover, the relative distance of the cities (or length of the edges of the graph), which is providing the heuristic information and also contributing to the objective function, does not depend on the previous cities visited during the tour. This means that an edge will give the same contribution to the overall length of the tour, regardless of the rest of the tour. For example, Fig. 5.11 shows two different tours for the given TSP instance: 1-4-3-2-5 (continuous line) and 1-2-4-3-5 (dashed line). The edge 3-4 is shared by the two different tours: this edge will obviously contribute in the same way to the objective function in both tours, i.e. the distance between city 3 and 4. This is due to the fact that the objective function is the total distance covered by the tour.

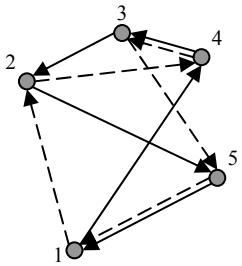


Fig. 5.11. A five-node instance of the TSP, with two possible solutions, identified by continuous and dashed arrows.

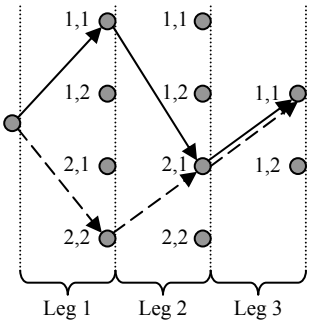


Fig. 5.12. A representation of a three-leg MGA problem, and two proposed solutions, identified by continuous and dashed arrows. Each node represents a combination of body/type of transfer. Despite the two solutions share the same parameters for the last leg ([1, 1])

This is not true for the problem under consideration. Fig. 5.12 is a representation of a simple instance of the problem: it has 3 legs, 2 types of transfer for each leg, 2 planets for the swing-bys, and 1 target planet. Each node represents a possible planet in combination with a type of transfer. The couples of numbers next to each node in Fig. 5.12 are the two integers identifying the leg in the solution vector (see Section 5.2.1). A solution is generated by selecting one node for each leg, thus generating a tour which connects the starting node to one of the final nodes, representing types of transfers to reach the target planet. The figure represents two possible solutions to the problem: [1, 1, 2, 1, 1, 1] (continuous line) and [2, 2, 2, 1, 1, 1] (dashed line). These two solutions share the same parameters for the last leg: [1, 1]. This means that they reach the same target planet with the same type of transfer. Because of the dependency of each leg from the initial conditions, it is not possible to state that the last leg is the same for the two solutions: in fact, the two trajectories can be consistently different, and lead to different final conditions and objective function.

For this reason, it makes no sense for example to assign a value to the set of parameters [1, 1] on leg 3 in Fig. 5.12. It is allowed instead to assign a value to the edge 3-4 in Fig. 5.11.

A different representation of the continuous-line solution in Fig. 5.12 is the one shown in Fig. 5.13. This other representation exploits a tree to underline that every branch of the tree depends on the previous ones. Both solutions of Fig. 5.12 are highlighted with arrows in the tree, and it is clear that the set of parameters [1, 1] for leg 3 leads to two different solutions.

5.3 Case Studies

The proposed optimisation method was applied to two case studies inspired by the BepiColombo [32] and Cassini [22] missions.

Since the launch date is not taken into account in the optimisation, in the following tests it is considered fixed. In a real mission design case, where the launch date is to be determined, the entire launch window can be discretised with a given time step, and a systematic scan of several dates within the whole launch window should be run. The launch direction φ_0 is also kept fixed in these tests, although it is easy to find heuristics for determining the value of this parameter, or discretise it and include it in the optimisation process as an additional variable.

ACO-MGA was tested against population-based algorithms, which are known to perform well on these kinds of problems. In particular, it was chosen to use the genetic algorithm implemented in MATLAB® within the Genetic Algorithm and Direct Search Toolbox™ (GATBX) [120], the Non-dominated Sorting Genetic Algorithm (NSGA-II) [121], and an implementation of Particle Swarm Optimization in Common type [122]. Settings for all the optimisers will be specified for each test case. While NSGA-II can deal with discrete variables, GATBX and PSO only use real variables: a wrapper of the objective function was coded to round the continuous solution vector to the closest integer.

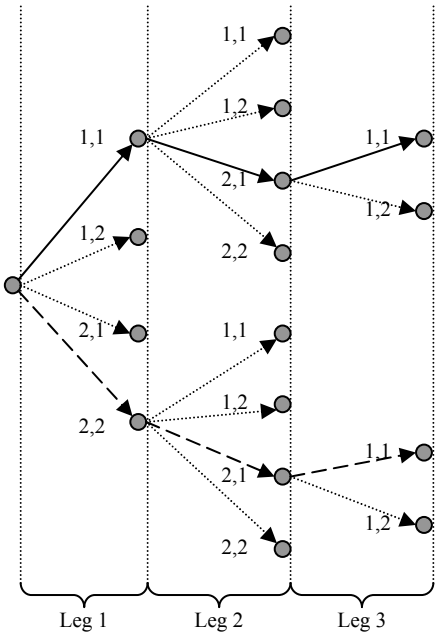


Fig. 5.13. Tree representation of the MGA problem of Fig. 5.12, and the two proposed solutions. This representation highlights the dependence of each edge of the graph from the previous history.

Due to the stochastic nature of the methods involved in the comparative tests, all the algorithms were run 100 times. The performance index used to compare the ACO-MGA against the other global optimisers is the success rate: according to the theory developed in [65] 100 repetitions give an error in the determination for the exact success rate of less than 6%.

Some preliminary tests showed that the best performances of ACO-MGA are achieved if the algorithm is run in 2 steps, using different sets of parameters. In particular, in the first step, the weights w_{planet}, w_{type} are set to 0. Remembering Eq. (5.9) this choice translates into an initial pure random search. In fact, the feasible solutions found do not alter the probability distribution. On the other hand, the probability of taboo partial solutions is still set to zero to avoid their re-exploration. In the second step, weights are set to non-null values, to explore around the feasible solutions found. The values of w_{planet}, w_{type} are set such that:

$$w_{planet}, w_{type} = \tilde{w} \cdot y_{est} \tag{5.10}$$

where y_{est} is the typical, estimated value for the objective function, and it is used to normalise the right hand side of Eq. (5.9). In this way, choosing for example $\tilde{w} = 1$, means that 1 is added to the pheromone of a given element every time a matching solution with objective y_{est} appears in the feasible list. Obviously the value of

added pheromone is higher if the objective value of the matching feasible solution is lower.

This two-step procedure can be explained in the following way. The first step allows a random sampling of the solution space, with the aim of finding a good number of feasible solutions. This is done to prevent the algorithm to stagnate around the first feasible solution found.

The second step intensifies the search around the feasible solutions which were found at step one. Because of Eq. (5.9), feasible solutions with low objective value are likely to be investigated more. In addition, the random component in the process does not forbid to keep exploring the rest of the search space.

The test cases were run on an Intel® Pentium® 4 3 GHz machine running Microsoft® Windows® XP.

5.3.1 BepiColombo Case Study

In this mission, the spacecraft departs from Earth (on 15 August 2013, i.e. $t_0 = 4974.5$ d, MJD2000) to reach a scientific orbit around Mercury: therefore, it is advisable to minimise the relative velocity at arrival v_∞. The launch date has been set to match the one proposed in [31], such that a reference solution for this date is available. Three legs (and thus two swing-bys) are considered for the transfer, while the launch angle was set to $\varphi_0 = \pi$. For the first and second legs, the following set of parameters was considered:

$$i = 1,2: \begin{cases} \mathbf{q}_{P,i} = \{\text{Mercury, Venus, Earth}\}; \\ \mathbf{q}_{1,i} = \{0\}; \\ \mathbf{q}_{2,i} = \varnothing; \\ \mathbf{q}_{3,i} = \{1,2,3,4\}; \\ \mathbf{q}_{4,i} = \varnothing; \\ \mathbf{q}_{5,i} = \{0,1\} \end{cases}$$

Since there is no DSM, the parameters $n_{rev,1}$ and $f_{p/a}$ are not used, and thus are set to 0. The number of revolutions is entirely controlled by the parameter $n_{rev,2}$. In general, there is no easy way to identify whether the first or the second orbital intersection is the best one, so the binary parameter $f_{1/2}$ was left free to be chosen by the ants. For the third leg, parameters were chosen from:

$$\mathbf{q}_{P,3} = \{\text{Mercury}\};$$
$$\mathbf{q}_{1,3} = \{-50, 0, 50\} \text{ m/s};$$
$$\mathbf{q}_{2,3} = \{0\};$$
$$\mathbf{q}_{3,3} = \{1, 2, 3, 4\};$$
$$\mathbf{q}_{4,3} = \{0, 1\};$$
$$\mathbf{q}_{5,3} = \{0, 1\}$$

In this case, a DSM can be exploited to target Mercury and to reduce the v_∞ with respect to it. The departure excess velocity module v_0 is constrained to be within 2 and 4 km/s, which implies the following set of starting guess points for Brent's method:

$$[2, 2.5, 3, 3.5, 4] \text{ km/s}$$

The following set of starting points for r_p was used instead for both Venus and Mercury:

$$[1, 1.02, 1.04, \ldots, 5] R_P$$

and $r_{ps} = \{r_p, -r_p\}$. The total time of flight was limited to a maximum of 10 years and the objective function for a complete solution is the v_∞ at Mercury. The average time for evaluating one solution (finding all the trajectories that it generates) of this test case is 1.75 ms, and there are 54000 distinct solutions. Thus, a systematic approach, scanning all the solutions, would require about 94.5 s.

All the optimisers were run for up to 2000 function evaluations. GATBX, NSGA-II and PSO were run with the settings shown in Table 5.3. In addition, the initial population of GATBX is spread in the whole domain.

Results in the form of statistical parameters over the 100 runs are presented in Table 5.4. The best average value is computed on the runs which found feasible solutions only, while the value of 6 km/s as a target value for the v_∞ has been chosen to compute the success rate according to the procedure proposed in [31].

Table 5.3. Parameters of GATBX and NSGA-II for the BepiColombo test case.

GATBX		NSGA-II		PSO	
Parameter	Value	Parameter	Value	Parameter	Value
Generations	20	*ngen*	100	*iter*	20
PopulationSize	100	*popsize*	20	*pop*	100
StallGenLimit	+∞	*pcross_bin*	0.5	*iiw*	0.9
		pmut_bin	0.5	*fiw*	0.4

Table 5.4. Comparison of the performances of ACO-MGA, GATBX, NSGA-II on 100 runs of the BepiColombo problem. Refer to the text for the two settings of ACO-MGA.

	Average best value found, km/s	% runs finding < 6 km/s	% runs finding a feasible solution
ACO-MGA (1)	6.102	26%	100%
ACO-MGA (2)	6.176	24%	100%
GATBX	8.82	13%	98%
NSGA-II	9.957	2%	100%
PSO	11.443	3%	99%

Two different set of parameters were used for ACO-MGA: (1) in Table 5.4 refers to a first step with 200 iterations and $w_{planet}, w_{type} = 0$, followed by a second step of 200 iterations with $w_{planet}, w_{type} = 20 \cdot 3$ km/s; (2) instead uses the same values for the weights, but the first step has only 60 iterations, while the second 440. This shows that a high number of iterations allocated to the second step does not necessarily improve the results, if the first step did not sample enough random points in the solution space. ACO-MGA was run on the same optimisation problem for different values of the weights, and the best results were obtained with the values mentioned before. Because of the normalisation shown in Eq. (5.10), the weight values appear to have general validity, and can be applied also to other transfer problems, as will be shown in the next case study.

Since the size of the population is very important for genetic-based algorithms, and it can affect the results significantly, this case study was also run 100 times with a population of 40 and 100 individuals (maintaining the predefined number of total function evaluations by varying the number of generations accordingly). For NSGA-II, it resulted that there was no noticeable change in the quality of the results over 100 runs. This is related to the fact that NSGA-II is not completely converging with 2000 function evaluations.

PSO was run for a number of different settings, ranging between $[iter = 10, pop = 200]$ and $[iter = 200, pop = 10]$. Very little difference in the quality of the results was found, being the best choice close to the former set of parameters; hence the choice in Table 5.3, that was used for obtaining the results in Table 5.4. As a comparison, by choosing the latter set of parameters, an average of about 14 km/s is obtained, with only 75% of feasible runs and 1% below 6 km/s.

As a reference, by increasing the number of function evaluations to 10000, the average value of the solution from NSGA-II lowered to 7.47 km/s, with 18% below 6.5 km/s. This shows a considerable improvement but the result is still worse than ACO-MGA.

For GATBX, instead, results were changing significantly, and the settings leading to the best solutions were used.

The results in Table 5.4 point out that, while all the algorithms find feasible solutions in practically all the runs, the quality of the solution is much better for ACO-MGA, for either of the two different sets of settings used. Moreover, NSGA-

II found a good solution only in 2 runs, and GATBX in 13. ACO-MGA, instead, found a good solution in about 25% of the runs.

The run time for ACO-MGA (1) was about 220 s. The simplicity of the test case, together with the implementation of ACO-MGA in a interpreted language like MATLAB®, makes the use of an optimisation method slower than the systematic scan of the whole solution space. Note that this will not happen in the more complex Cassini test case.

The performance values of ACO-MGA can be greatly improved by considering a slightly different instance of the BepiColombo transfer problem. Let us assume a list for r_p which spans down to $0.9\,R_P$:

$$\left[0.9, 0.92, 0.94, \ldots, 5\right] R_P$$

Here a comment is needed: it is usual practice, in preliminary mission design, to consider a safety margin in the closest approach of a planet during a swing-by manoeuvre. Usually this translates in considering a radius of pericentre not smaller than $1.1\ R_P$. The main reason for using a slightly lower minimum altitude is that the number of feasible solutions in the solution space increases, since each swing-by can provide a higher deflection of the velocity vector, and thus the search for the optimal solution results more favourable for all the optimisers, but in particular for the population-based ones. We can also assume that the safety margin can be added when the solution is re-optimised with a more complete model, and assume that a DSM can compensate for the lack of swing-by deflection. Note that this trick will not be adopted in the more realistic case study presented in the next section.

The results for this second test case are presented in Table 5.5. All the algorithms perform better on this second instance of the problem, but ACO-MGA improved dramatically, finding good solutions in 98% of the runs. The best solution found by ACO-MGA is represented in Fig. 5.14. The sequence for this solution is EVVMe, and the objective value, i.e. the final relative velocity, is $v_\infty = 5.5082$ km/s. The parameters for this solution are presented in Table 5.6.

Table 5.5. Comparison of the performances of ACO-MGA, GATBX, NSGA-II on 100 runs of the BepiColombo problem with extended bound on r_p.

Optimiser	Average best value found, km/s	% runs finding < 6 km/s	% runs finding a feasible solution
ACO-MGA (2)	5.67	98%	100%
GATBX	8.15	26%	97%
NSGA-II	9.58	7%	100%
PSO	11.32	5%	97%

Table 5.6. Parameters of the best solution found by ACO-MGA for the BepiColombo case study.

Parameter	Leg 1	Leg 2	Leg 3
Planet	V	V	Me
m_{DSM} , m/s	0	0	-50
$n_{rev,1}$	0	0	0
$n_{rev,2}$	1	3	4
$f_{p/a}$	0	0	1
$f_{1/2}$	0	0	1

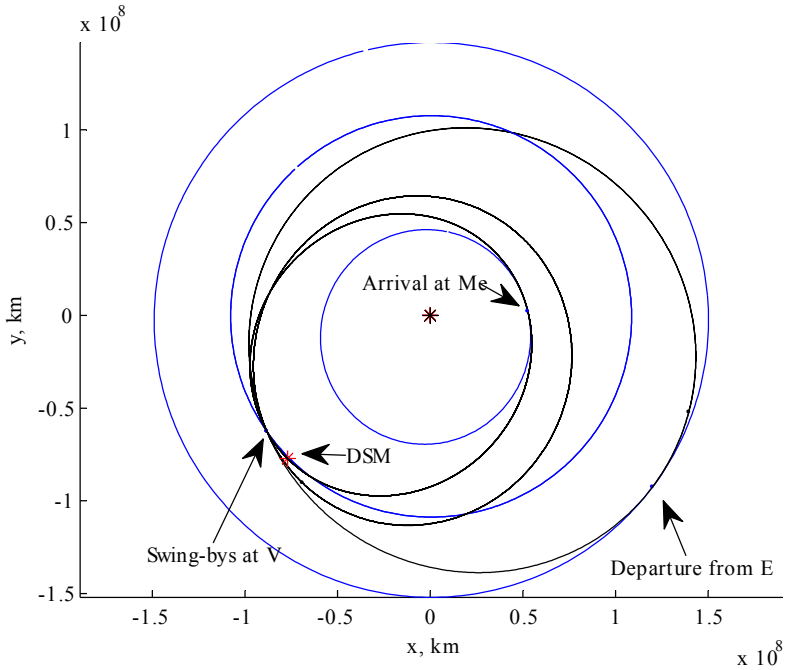
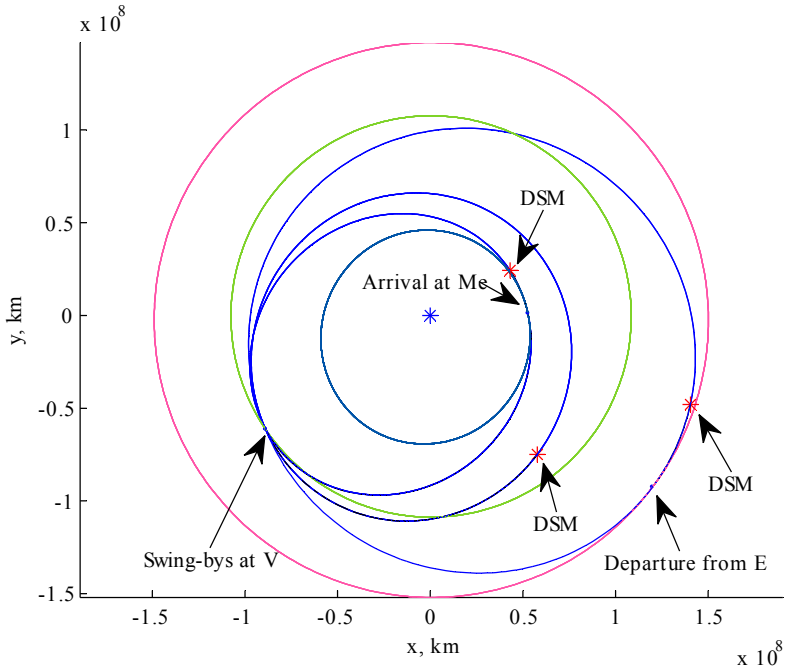


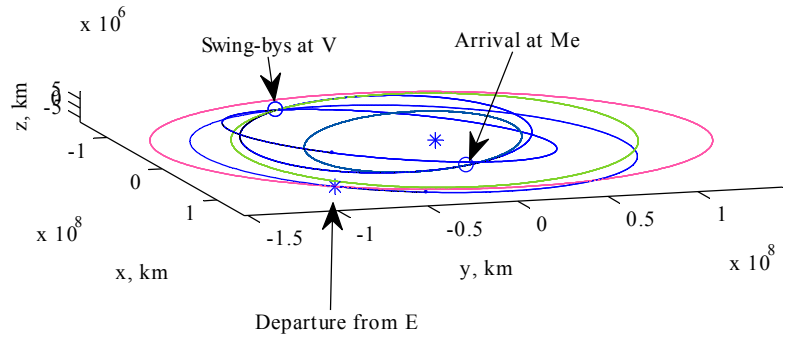
Fig. 5.14. The best solution found by ACO-MGA, with objective value 5.5082 km/s. The planetary sequence is EVVMe.

The validity of the model is verified by re-optimising the best solution found with ACO-MGA, but using a full 3D model with 1 free DSM per leg [102], and minimising the total Δv. The resulting trajectory is shown in Fig. 5.15. In particular, Fig. 5.15 (a) shows the x-y projection of the trajectory, while Fig. 5.15 (b) shows the relevant out of plane component that is introduced while considering a full 3D model with real ephemerides, and mainly due to the high inclination of Mercury.

Finally, Table 5.7 compares some parameters of the 2D solution with the re-optimised 3D solution and the ESOC reference solution. Note the similarity of the 3D trajectory to the one found by ESA and described in [31]. The values α_i refer to the fraction of time of flight T_i of i^{th} leg at which the DSM occurs in the 3D model.



(a) Projection on ecliptic plane



(b) Side view

Fig. 5.15. The solution re-optimised with a full 3D model, minimising the total Δv. (a) The projection of the trajectory on the ecliptic plane; (b) The side view highlights the out-of-plane components of the trajectory, mainly due to the need of targeting Mercury, whose orbit is relatively high inclined.

It is interesting to note that the launch velocity is higher in the 3D version mainly because of the small inclination of the planets. The v_∞ also is higher due to the inclination of the target planet but again the difference is small.

On the other hand, the optimisation is able to move and fine tune the position of the DSMs along the transfer leg, thus reducing the amount of deep space Δv. This is particularly evident on the third leg, in which the position of the DSM

(represented by α_3) was considerably moved forward in time. This can only partially be done by the planar model, as DSMs are constrained at the apsides of the orbit. Note that the values of α_i for the ACO-MGA model are not parameters, but were computed *a posteriori*.

Table 5.7. Characteristics of the best solution found by ACO-MGA, the same solution after optimisation with a full 3D model, and the ESOC reference solution.

Variable	ACO-MGA	3D optimised	ESOC
v_0, km/s	3.63	3.8	3.79
Δv_1, m/s	0	7	7
Δv_2, m/s	0	0	0
Δv_3, m/s	50	11	11
v_∞, km/s	5.51	5.68	5.68
T_1, d	438.5	438.5	438
T_2, d	674.1	674	674.1
T_3, d	630.8	630.8	630.9
α_1	0.04	0.04	0.05
α_2	0.02	0.02	0.1
α_3	0.01	0.12	0.12
$r_{p,1}$, planet radii	1.47	1.3	1.3
$r_{p,2}$, planet radii	2.14	1.21	1.21

LAUNCH DATE ANALYSIS

As mentioned before, the algorithm, at the current state, does not perform any kind of search on the launch date t_0. In fact, this variable is not even included in the solution vector **s**. Rather, if the launch date is not fixed, but a launch window is available, a systematic scan can be performed to find the best launch date, and the corresponding solutions. This procedure is not always applicable: in fact, if re-running the algorithm for a small change in the launch date, the solutions that ACO-MGA finds are substantially different, then the systematic scan along t_0 is not feasible, and this variable must be taken into account in the optimisation process. If, on the other hand, a small displacement along t_0 causes a small change in the best solution found (e.g. same planetary sequence, possibly different types of transfers, similar objective value), then the systematic scan is a tool for identifying the promising launch possibilities.

A test for verifying this assertion was run using the BepiColombo transfer problem. As stated before, the optimal launch date can be found in [31], and let identify it here with $t_0^* = 4974.5$ d, MJD2000.

Five different launch dates, in a window of 10 days around the one chosen by ESA, were considered, and for each one of them, 100 runs were used. The corresponding best solution values are found in Table 5.8. The result is that the best

solution is found about 1 day before t_0^*, while earlier or later launches become less convenient. In addition, all the solutions around the ideal launch date have the same planetary sequence of swing-bys.

The discrepancy between the value of v_∞ found by ACO-MGA and the one in [31] has two causes: the first is that ACO-MGA does not take into account the inclination of the planets, and the orbit of Mercury is highly inclined. The second is that the ESA solution was found as a part of a longer trajectory, and thus with a different objective.

The same reasons explain why, according to ACO-MGA, the ideal launch date is 1 day earlier. As a matter of fact, this is not a problem, and a subsequent local optimisation of the ACO-MGA solutions with a full model would tune the launch date.

The same analysis was extended for a wider interval of $\left[t_0^* - 30\text{ d}, t_0^* + 30\text{ d}\right]$, with a one-day step, and 100 runs for each t_0. Fig. 5.16 shows, for each launch date, the best and average solution found by the 100 runs: it highlights that there are two intervals of dates in which the optimal solution is more difficult to find by the ants. The same information can be obtained more in detail from Fig. 5.17, that displays the number of runs that found a solution at most 0.01, 0.1 and 0.5 km/s worse than the best one for that date. The same bar plot highlights that all the runs found a feasible solution.

This test demonstrates that the 2D model and the search process in ACO-MGA yield a distribution of solutions, along the departure time axis, that is consistent with the existing reference solution for this mission. The convergence of ACO-MGA is robust against variations of the launch date, as it consistently provides, solutions with unchanged planetary sequence that display a small variations in the cost function for small variations of the launch date. These properties together with the small computing time suggest that ACO-MGA can be used to systematically scan the launch dates in search for an optimal one. Note that, as the model in ACO-MGA is intended for the generation of first guess solutions, the scan of the launch dates is expected to provide only an estimated location of the optimal point along the launch date coordinate.

Table 5.8. Best solutions to Mercury found by ACO-MGA for different launch dates.

Launch date t_0	Optimal sequence	$y \equiv v_\infty$, km/s
$t_0^* - 5$ d	EVVMe	5.98
$t_0^* - 1$ d	EVVMe	5.84
t_0^*	EVVMe	6.10
$t_0^* + 1$ d	EVVMe	6.62
$t_0^* + 5$ d	EVVMe	6.72

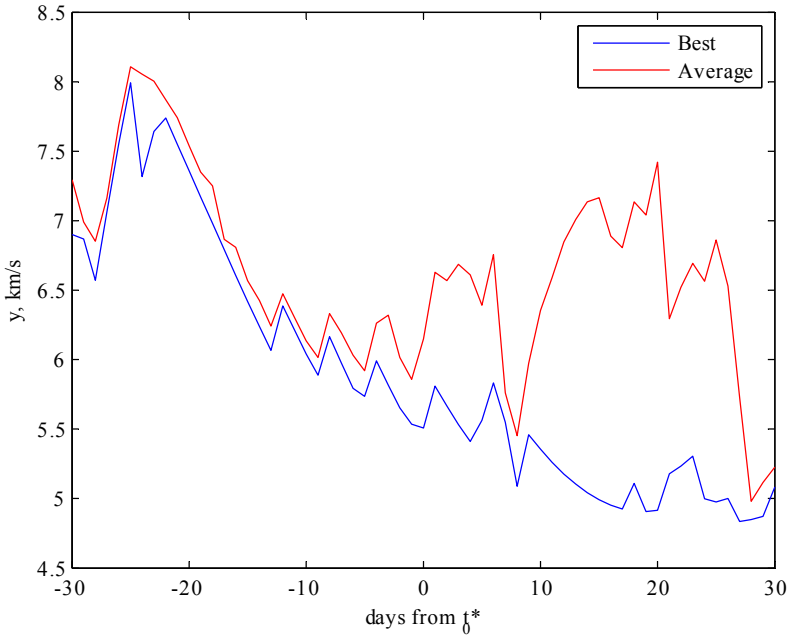


Fig. 5.16. Average and best solutions found by 100 runs for each launch date around the optimal one.

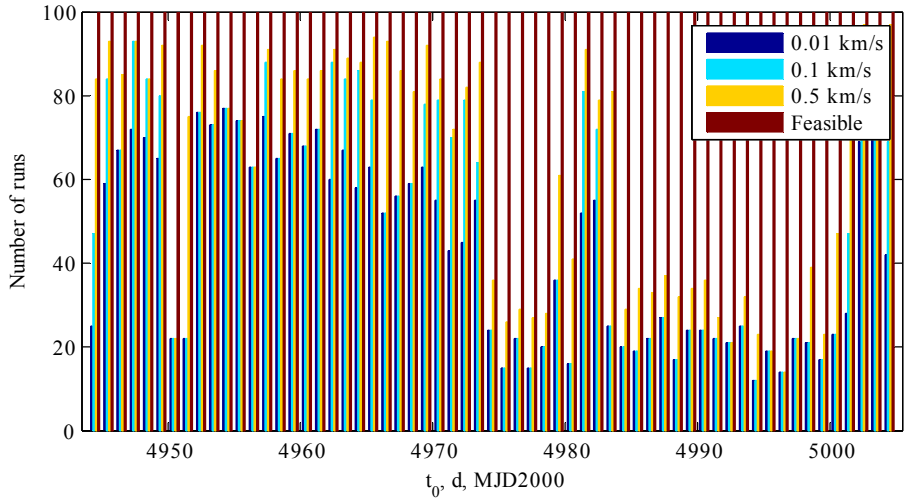


Fig. 5.17. For each launch date, number of runs (out of 100) that found a solution which is at most 0.01, 0.1 and 0.5 km/s worse than the best solution found for that launch date. Also, runs that found a feasible solution.

5.3.2 Cassini Case Study

Cassini is the ESA-NASA mission to Saturn. The planetary sequence designed for the mission, EVVEJS is particularly long, allowing a substantial saving of

propellant. For testing the ACO-MGA we will make use of a 5-leg trajectory, with starting date $t_0 = -779$ d, MJD2000, corresponding to 13 November 1997. The following sets of parameters were used for the first 3 legs:

$$i = 1,2,3: \begin{cases} \mathbf{q}_{1,i} = \{\pm 600, \pm 350, \pm 200, 0\} \text{ m/s} \\ \mathbf{q}_{2,i} = \{0\} \\ \mathbf{q}_{3,i} = \{0\} \\ \mathbf{q}_{4,i} = \{0,1\} \\ \mathbf{q}_{5,i} = \{0,1\} \end{cases}$$

Thus giving quite a wide choice of DSMs. For the last two legs, instead, no DSM is allowed:

$$i = 4,5: \begin{cases} \mathbf{q}_{1,i} = \{0\} \\ \mathbf{q}_{2,i} = \varnothing \\ \mathbf{q}_{3,i} = \{0\} \\ \mathbf{q}_{4,i} = \varnothing \\ \mathbf{q}_{5,i} = \{0,1\} \end{cases}$$

The planets available for swing-bys are $\mathbf{q}_P = \{\text{Venus, Earth, Jupiter}\}$, while the target planet is obviously fixed to Saturn. The number of maximum full revolutions was fixed to 0, as it can be seen from the choice of parameters $n_{rev,1}$ and $n_{rev,2}$. This is done to limit the total time of flight of the mission. Since the trajectory is going outwards of the orbit of the Earth, every full revolution implies more than one additional year in the transfer time. The main aim of this case study, then, is to assess the ability of finding promising planetary sequences, using DSMs. The total number of distinct solutions for this test is 7,112,448, and the average time to evaluate a solution is 1.26 ms. This translates in 8961.7 s (or about 2.5 hours) to systematically evaluate all the solutions.

As for BepiColombo, the launch excess velocity module was bounded between 2 and 4 km/s. For the swing-bys of Earth and Venus, the radii of pericentre are

$$[1.1, 1.2, 1.3, \ldots, 5]\, R_P$$

while a different choice was adopted for Jupiter. In fact, the mass of this planet is considerably bigger than the masses of Venus and Earth, so higher radii of pericentre are enough to achieve considerable deviations. Furthermore, since the function $\Delta\theta(r_{ps})$ is very smooth in this case, the first guesses are spaced with a 5 Jupiter radii step size:

$$[5, 10, 15, \ldots, 100]\, R_P$$

Regarding the choice of the objective function, it has to be noted that for all the missions to outer planets, the time of flight becomes very important, as very long missions are needed to reach farther destinations. Even limiting the number of complete revolutions to zero, is not enough to guarantee a mission with reasonable duration. Therefore, it is important to include the total time of flight T in the objective function, in addition to the total Δv. Since the current algorithm cannot deal with multi-objective optimisation, the total time of flight and the v_∞ are weighed inside the objective function in the following way:

$$y = v_\infty + \beta T$$

and for this test case the weight on T was chosen to be $\beta = 1/1000$ km/s/d.

The total time of flight has been limited to a maximum of 100 years. This bound may seem to be too high, since a realistic time span of a transfer to Saturn is around 10 years. However, the model considers unfeasible all the solutions longer than the specified time of flight threshold, and the optimiser saves them as taboo. Therefore, limiting the time of flight to lower values would over-constrain the search for optimal solutions. Better results are obtained by allowing long solutions to be returned as feasible, and introducing their duration into the objective function.

The three optimisers were run at first for 4000 and then for 6000 function evaluations. The weights of ACO-MGA were set to $w_{planet}, w_{type} = 0$ for the first step, and $w_{planet}, w_{type} = 20 \cdot 7$ km/s for the second step. For each step, the number of iterations of ACO-MGA was set such that the expected total maximum number of function evaluations was 4000 for the first bunch of runs and 6000 for the second bunch of runs. In particular, for 4000 function evaluations, the number of iterations allocated to the first step was 200 and the number of iterations allocated to the second step was 300. For 6000 function evaluations, instead, the number of iterations allocated to the first step was increased to 600 and the number of iterations allocated to the second step was left equal to 300. With these settings, a run of ACO-MGA requires, on average, 1900 function evaluations and 161 s, if the upper limit is 4000, and 3300 function evaluations and 273 s, if the upper limit is 6000 evaluations. This is considerably faster than the exhaustive scan of the solution space and the number of function evaluations is lower than in the case of GATBX and NSGA-II, however the computational time for each single run of GATBX, which is fully coded in MATLAB®, is, on average, half of the one required to ACO-MGA as the access to the taboo and feasible lists is currently not optimised. NSGA-II is even faster as the code is fully in C. Therefore, a fair comparison would allow GATBX and NSGA-II to perform a higher number of function evaluations. However, a run of ACO is considerably faster than the exhaustive scan of the solution space.

The parameters used for GATBX and NSGA-II are reported in Table 5.9. The comparative results for the two sets of runs are shown in Table 5.10. It can be seen that, for 4000 evaluations, ACO-MGA found feasible solutions in 91% of the runs, compared to 25% of GATBX and 26% of NSGA-II. The average ACO-MGA solution is also slightly better than GATBX, and considerably better than NSGA-II.

The performances of ACO-MGA increase significantly by using 6000 evaluations: all the runs produce a feasible solution, and in 80% of the cases, the best solution found is below 16 km/s. The average value of the solution also decreases to 15.434 km/s. It is interesting to note that, for GATBX, the average best solution found with 6000 evaluations is higher than for 4000: this is partly balanced by the fact that it finds feasible solutions in 28% of the runs. Another thing worth noticing is that NSGA-II finds more often feasible solutions than GATBX, but their quality is on average worse.

The best solution found through ACO-MGA has an objective value of 6.9686 km/s, corresponding to the parameters shown in Table 5.11. The characteristics of this solution can be found in Table 5.12, compared to the best solution known so far as reported in the ACT web site. The trajectory of the ACO-MGA solution is shown in Fig. 5.18, while the 3D reference solution is in Fig. 5.19.

It is interesting to sort the feasible sequences found by ACO-MGA according to the best objective value that they can achieve. The bar graph in Fig. 5.20 shows the outcome: note that every sequence has a trajectory associated to it, modelled as shown before, and thus taking into account the phasing problem. This means that these solutions could be re-optimised with a more detailed model (in particular including the third dimension), leading to actual transfer solutions.

Table 5.9. Parameters of GATBX and NSGA-II for the Cassini test case.

Function evaluations	GATBX Parameter	GATBX Value	NSGA-II Parameter	NSGA-II Value	PSO Parameter	PSO Value
	StallGenLimit	$+\infty$	*pcross_bin*	0.5	*iiw*	0.9
			pmut_bin	0.5	*fiw*	0.4
4000	*Generations*	20	*ngen*	200	*iter*	200
	PopulationSize	200	*popsize*	20	*pop*	20
6000	*Generations*	30	*ngen*	300	*Iter*	300
	PopulationSize	200	*popsize*	20	*Pop*	20

Table 5.10. Comparison of the performances of ACO-MGA, GATBX, NSGA-II on 100 runs of the Cassini problem.

Function evaluations	Optimiser	Average best value found, km/s	% runs finding < 16 km/s	% runs finding a feasible solution
4000	ACO-MGA	16.24	44%	91%
	GATBX	16.349	14%	25%
	NSGA-II	20.426	5%	26%
	PSO	24.93	1%	3%
6000	ACO-MGA	15.434	80%	100%
	GATBX	16.526	17%	28%
	NSGA-II	20.122	7%	37%
	PSO	18.1334	1%	14%

Table 5.11. Parameters of the best solution found by ACO-MGA for the Cassini case study.

	Leg 1	Leg 2	Leg 3	Leg 4	Leg 5
Planet	V	V	E	J	S
m_{DSM} , m/s	600	-350	0	0	0
$n_{rev,1}$	0	0	0	0	0
$n_{rev,2}$	0	0	0	0	0
$f_{p/a}$	0	1	1	0	0
$f_{1/2}$	0	1	0	0	1

Table 5.12. Characteristics of the best solution found by ACO-MGA and the reference solution for the Cassini case study.

Variable	ACO-MGA	Reference
v_0 , km/s	3.14	3.259
Δv_1 , m/s	600	480
Δv_2 , m/s	350	398
Δv_3 , m/s	0	0
Δv_4 , m/s	0	0
Δv_5 , m/s	0	0
v_∞ , km/s	4.21	4.246
T_1 , d	168	167
T_2 , d	423	424
T_3 , d	53	53
T_4 , d	596	589
T_5 , d	2290	2200
α_1	0.83	0.77
α_2	0.52	0.51
α_3	0.16	0.02
α_4	0.02	0.26
α_5	0.13	0.6
$r_{p,1}$, planet radii	1.61	1.34
$r_{p,2}$, planet radii	1.25	1.05
$r_{p,3}$, planet radii	1.32	1.30
$r_{p,4}$, planet radii	68.3	69.8

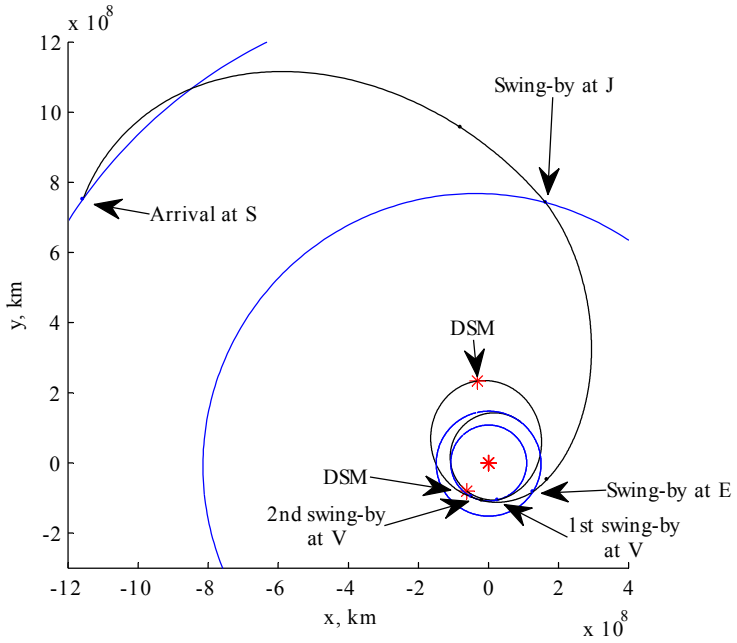


Fig. 5.18. Solution to the Cassini problem found with the 2D model and using ACO-MGA.

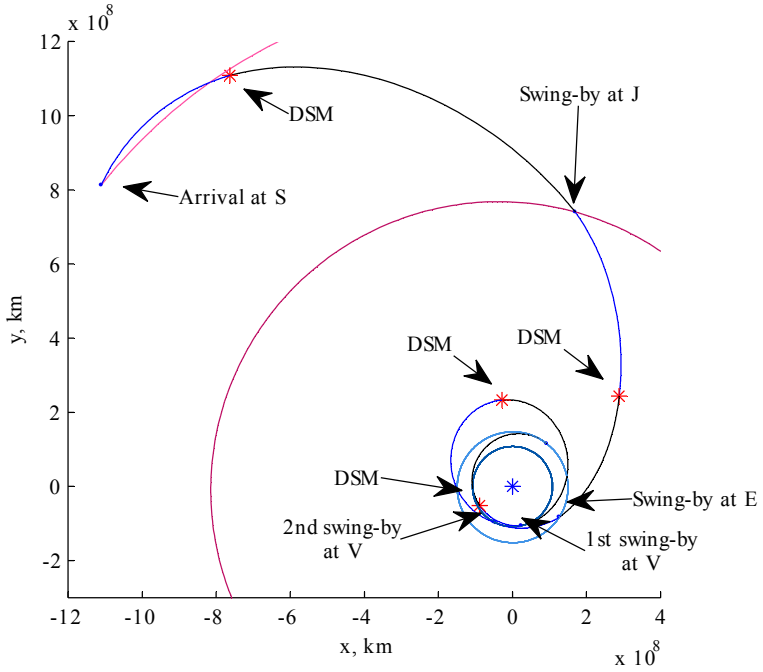


Fig. 5.19. Cassini reference solution found for the global trajectory optimisation problem proposed by ESA ACT.

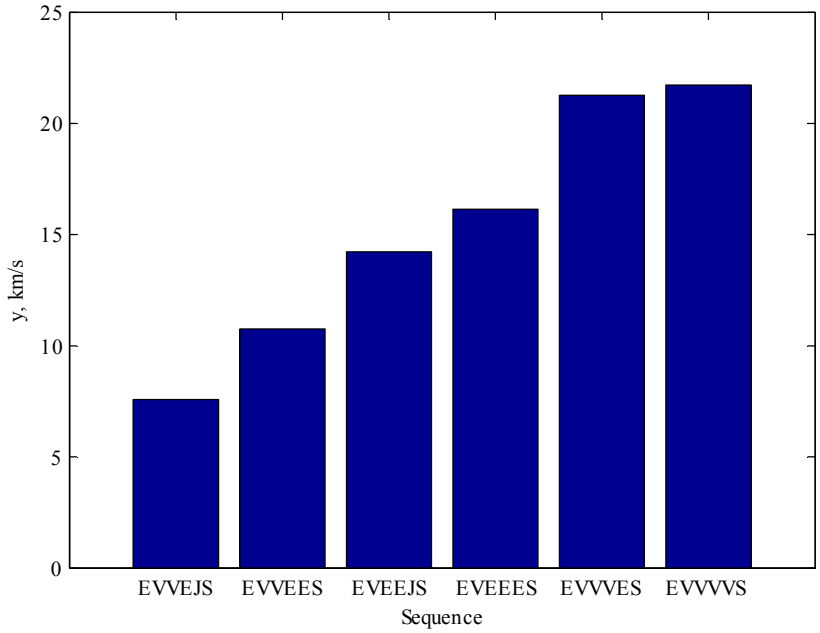


Fig. 5.20. Best objective values found for each sequence. All the other sequences are either unfeasible or with a very high objective.

5.4 Summary

This chapter introduced a novel formulation of the automatic complete trajectory planning problem and proposed a new algorithm (ACO-MGA), based on the ant colony paradigm, to generate optimal solutions to this problem. Each solution is a complete, scheduled plan. A specific trajectory model was developed to efficiently generate families of scheduled trajectories for multi-gravity assist transfers, once a plan is available. The 2D trajectory model proved to be accurate enough to closely reproduce known MGA transfers even with moderate inclinations. Furthermore, the scheduling of the trajectories is fast and reliable allowing for the evaluations of thousands of plans in a short time.

ACO-MGA operates an effective search in the finite space of possible plans. The algorithm demonstrated a remarkable ability to find good solutions with a very high success rate, outperforming known implementations of genetic algorithms.

6

CONCLUSIONS

This dissertation presented two global optimisation techniques for multiple gravity assist (MGA) trajectories with deep space manoeuvres (DSM). This final chapter will draw some conclusions on this study. After a summary of the work, some remarks will be given, including a summary of the results, followed by the limitations and possible areas of further investigation.

6.1 Summary of the Work

In this study, tools and methods for interplanetary trajectory design were investigated, with particular focus on MGA missions. Gravity assists are necessary to deliver consistent payloads to high-energy targets (like farther planets or highly inclined orbits) with a limited amount of propellant. High-thrust, impulsive Δv trajectories were investigated. The research focused mainly on two aspects: the development of a tool for automated trajectory design and the identification of families of solutions in the neighbourhood of locally optimal ones. Automated design refers to the ability to reliably produce good solutions for a wide range of cases, with minimum human intervention. This is a very desirable feature that can answer to the main requirement of the modern approach to space mission design: the identification of many different optimal or sub-optimal trajectories to be used in the initial trade-off analysis. The author believes that the tools developed in this thesis satisfy this requirement. It was also highlighted that the design of MGA trajectories consists of two problems: the first is the identification of a planetary sequence, and the second is to find optimal solutions for a given sequence. If these two problems are solved at different stages, then the solution approach has two-levels. If instead the two problems are solved at the same time, for example with a hybrid optimisation method, then the approach is said to be integrated. This thesis investigated both approaches.

Chapter 2 introduced two models for MGA trajectories with impulsive DSMs. One, named position formulation, allows accommodating multiple DSMs for each

interplanetary leg, and needs a powered swing-by to match two consecutive legs. Each manoeuvre is determined by its position and timing, therefore the name of position formulation. A consequence of this choice is that each ballistic arc can be computed independently of the others. This model, thus, uncouples the planet-to-planet legs, at the cost of having to specify the position of each manoeuvre. A second model (velocity formulation) computes each leg by propagating the initial velocity, which is known from the previous leg, hence the name of velocity formulation. In this formulation, the legs are computed sequentially, and one leg depends on all the previous ones. On the other hand, swing-bys are unpowered and only the timing of the DSM is to be specified. Appendix B introduced a third, more versatile, trajectory model paradigm, to compose a trajectory with basic building blocks that represent events. This model is suitable for automatic definition of the types of events in a trajectory. The two previous models can be obtained using building blocks.

The model based on the velocity formulation was used to create a tool that solves the transfer design problem with a two-level approach (see Chapter 3). The method is based on the simple consideration that if a given transfer is infeasible (due, for example, to a high Δv) at the first leg, then there is no need to add and compute any more legs: that set of parameters can be removed from the solution space. If feasible sets are identified for the first leg, then the rest of the space can be pruned out. When the second leg is added to the trajectory, only a small part of the space has to be considered. The procedure continues, adding one leg at a time and building the whole trajectory incrementally (hence the name). Different techniques were proposed to search for feasible solutions and identify the feasible sets at each level.

Although the incremental pruning technique exploits the particular structure of the MGA global optimisation problem, the same problem can be (and in literature has been) tackled with off-the-shelf optimisers. Therefore, the incremental pruning was compared with other stochastic and deterministic global optimisation techniques (Multi-Start, Differential Evolution, Monotonic Basin Hopping, DIRECT and Multilevel Coordinate Search).

The incremental pruning procedure, like all two-level approaches, requires a planetary sequence defined in advance. To this aim, an algorithm for generating possible sequences was designed and implemented. Heuristic rules are used first to remove, from the tree of all possible planetary sequences, the trivial ones. A successive step exploits energetic considerations to find feasible sequences and rank them. This is done relying on some assumptions like circular and planar orbits, ballistic transfers, and neglecting phasing. The result is a limited list of feasible sequences ranked by value of infinite velocity at the target planet. These lists are then passed to the incremental pruning process to find optimal trajectories.

The combination of the sequence generator and the incremental pruning was used to solve real-world mission design cases. This was the topic of Chapter 4. The ESA missions BepiColombo and Laplace (currently in design phase) were used to this aim. Given initial conditions, requirements and constraints, interplanetary transfers were found with the tools proposed in this work, and compared with baseline solutions. For both the missions, a transfer phase and a resonant swing-by

phase were considered. The solutions found through the incremental method demonstrated to be similar, and in some cases better, than those selected for the respective missions as baseline. These test cases showed that the incremental pruning can be used not only to minimise the cost of a transfer, but also to meet other mission requirements, like the time of flight or aim at particular final conditions. This was done with the introduction of specific pruning functions. Moreover, it was shown that the feasible sets of optimal solutions found by the incremental pruning are suitable for further studies and to better understand the MGA problem under consideration. It was also verified that the same technique can be applied to the very special MGA transfers that use resonant swing-bys of the same planet to change the orbital parameters.

Finally, an integrated approach for MGA trajectory design was presented in Chapter 5. A specific planar trajectory model was developed, such that the phasing problem is solved internally, and the parameters to characterise the trajectory are reduced to a minimum. The real parameters are discretised, such that the whole trajectory model and sequence are represented only by categorical variables. Therefore, the automated design of an MGA trajectory is transformed into the search for the optimal path along an acyclic tree. The tree is explored by a population of virtual ants driven by heuristics inspired to the Ant Colony Optimization paradigm (hence the name ACO-MGA). The ants explore the tree of possible trajectories from root to leaves: when a feasible solution is found, it is stored in a list together with its objective value. Each ant chooses one branch of the tree probabilistically. The probability distribution is built and updated with the collection of all the feasible solution found by the ants: branches that led in the past to good solutions are more likely to be chosen. In addition, the algorithm keeps track of the solutions that are unfeasible at a given leg, and thus do not reach the solution. This way the ants avoid re-exploring unfeasible branches. ACO-MGA approach was tested on two different missions of increasing complexity: BepiColombo and Cassini. The performances were compared to other stochastic optimisers, on the same trajectory model. ACO-MGA outperformed all other global optimisers to which it was compared in the test cases, in terms of quality of the solution and repeatability of the results. It was also shown that this method allows a great time saving with respect to a systematic search in the tree, for complex problems like Cassini, in which millions of solutions exist.

6.2 Final Remarks

This research addressed the problem of automatic design of MGA trajectories, including the choice of the planetary sequence. The author believes that the tools and methods developed in this research will be useful for designing future space missions, as well as better understanding the complex nature of interplanetary trajectory design problems. The author also wishes that this study could contribute to and stimulate further research in this fascinating field.

All the results obtained with the methods proposed in this thesis were already discussed throughout the document, in particular in Chapters 3 and 4 for the two-level approach, and Chapter 5 for the integrated approach. In this section, a brief qualitative summary of the achievements of this study will be done. Then, some known limitations of the proposed solutions will be discussed. Addressing these limitations is what the author suggests as the next step of the research on this topic.

6.2.1 Fulfilled Objectives

The main objectives of this work can be summarised in two main points. The first is the automated preliminary design of MGA-DSM trajectories, including both the sequence of planetary swing-bys and the optimal path. Due to the complexity of the problem, systematic searches fail to provide solutions in a reasonable amount of time. Two stochastic approaches based on global optimisation are used, and their performances, in terms of speed, quality of the results and repeatability of the results, are proven through a set of test cases. These tools are intended to be used at the preliminary stage of the space mission design, when very little information usually is known about the transfer trajectory.

The second objective is the search of families of optimal or quasi-optimal solutions, as opposed to what is pursued in a standard global optimisation problem, i.e. the global optimum. This second point is fulfilled, by the two proposed methods, in two different ways: in the two-level approach, the incremental pruning identifies and preserves families of feasible solutions in clusters, and prunes out the rest of the search space. In the integrated approach, lists are used to store feasible solutions and to avoid re-exploration of unfeasible paths. This is not the case for other optimisers, like PSO, GA or DE: although population-based optimisers return a great number of solutions, these could be in fact quite similar when the optimisers reach convergence.

6.2.2 Other Major Achievements

All the methods were tested on case studies inspired by real missions, and they were able to find and reproduce well known optimal solutions. The incremental approach in particular was applied to real mission transfer trajectories that were under study, and in all the cases was able to replicate the baseline solutions and in some cases found better solutions that could be considered.

The incremental pruning is a tool that can be used at preliminary design stage, for assessing the transfer possibilities on a wide range of launch dates and parameters of the trajectory. However, it can be used also to investigate the solution space in the vicinity of a given solution, or for a limited set of launch dates and other trajectory parameters, for example when, at later design stage, a baseline mission is already identified, and backups are needed, or when some mission constraints are limiting the possibilities on the transfer trajectory.

One of the advantages of the incremental pruning is that it uses a three-dimensional trajectory model, including real planetary ephemerides, thus the phasing problem is solved exactly. The linked conic approximation demonstrated to

be very good for preliminary design stage, therefore the solutions obtained with the incremental process do not need to be further re-optimised.

Appropriate objective functions and pruning criteria were investigated and designed for specific transfer cases: targeting a given orbit, reducing the v_∞, as well as using resonant swing-bys. The test cases showed that the proposed criteria have general validity.

The boxes identified by the incremental pruning resulted to be a small fraction of the whole search space; therefore any subsequent study can focus on these small areas, in which optimal solutions are likely to be found, with great saving of computational effort. It was also shown that the feasible sets can be used for further studies and deeper investigation into the problem. Moreover, the pruning process produces, as an output, a number of locally optimal solutions.

The basic idea of discarding sets of solutions, based on some considerations on the problem, is somewhat similar to branch and bound techniques. In a classic branch and bound approach, only the best solution is to be found. Therefore, once a partial solution has an objective value that is higher than the one of the best full solution found so far, then that branch can be discarded. In a way, the pruning threshold is determined by the best solution so far. There are two issues in porting this methodology to the MGA problem. The first one is that the problem is not categorical, therefore it is not straightforward to define branches and to define upper and lower bounds for the objective function. The second major reason for which a branch and bound approach is not applicable is that we are interested in finding a wide set of feasible solutions, possibly including a high number of different mission options, and not only the global minimum. A classic branch and bound technique would discard a partial solution because it is slightly more expensive than the global best so far, but this solution could be interesting for a number of different reasons. The incremental pruning overcomes this limitation of the branch and bound approach through the clustering of feasible solutions and pruning the search space. Nonetheless, the incremental pruning finds a wide range of different solutions, as well as feasible sets, in which further investigation could be performed.

The integrated approach offers a solution for preliminary assessment of possible sequences for a given launch date. Although the trajectory model is less detailed than other models used in two-level approaches, it includes the ephemerides of the planets, and thus the phasing problem is taken into account. As a result, the outcome of this tool is a trajectory that is very similar to those found with more detailed models, and so a good candidate as a first guess for subsequent optimisation with a full model.

It is also important to underline that, despite this document presented the application to the interplanetary design problem, the original concepts and ideas behind the proposed approaches are independent from the specific application. The incremental pruning can be applied to the class of optimisation problems that can be decomposed into sub-problems, and through specific pruning criteria, tackled incrementally. The integrated approach based on ACO has applications on a number planning and scheduling problems: in particular, those the variants of the TSP in which the nodes are moving and the path cost depends on the whole history.

An example of these types of problems is the modified dynamic vehicle routing problem.

6.3 Current Limitations and Future Research

Despite the good results obtained with both the incremental approach and the ACO-based approach, there are some limitations in the present implementation. Here they will be discussed for both approaches, and some solutions are proposed for future work.

6.3.1 Incremental Pruning

The major issue with the incremental pruning technique is the choice of pruning thresholds. The author is aware that setting the pruning threshold is a critical point for the incremental process: a high threshold results in a weak pruning of the solution space, and possibly in an exponential growth of the number of boxes. On the other hand, an aggressive pruning may end up in losing good solutions. The rule of thumb that was used throughout the work is to limit the deep space Δv to a level that is achievable by the engine. This is the most conservative choice, as it preserves all the solutions that can theoretically be exploited by the spacecraft. However, it requires having an estimation of the performances of the engine and an estimation of the spacecraft mass. If these data are not available, then a more conservative choice shall be used. It was verified that even a conservative choice implies the pruning of a great part of the solution space. This is due to very high gradients in the vicinity of the local minima, which are a feature of the MGA trajectories. On the other hand, research needs to be done for automatic selection of the pruning thresholds, and criteria in general: heuristic rules should take into account the characteristics of the spacecraft, when available; otherwise, reasonable conservative choices should be made.

An area that was not deeply investigated in this research is an assessment of the optimisation methods to be used for identifying the feasible sets at each level. Although it was shown that a simple Multi-Start search within the incremental pruning can outperform other optimisers on the all-at-once problem, a deeper study can identify other optimisation techniques which are more efficient on this task. This could eventually further improve the performances of the incremental pruning.

Another partly open point of the incremental approach is related to the tuning of the clustering techniques. The incremental approach utilised clustering techniques to generate boxes that identify the feasible set. All the clustering techniques that were investigated require (at least) one parameter that has to be tuned. For example, for the Mean Shift clustering method, the parameter is the bandwidth; in the other methods, the number of subdivisions for each variable, and so on. In the present implementation, these parameters are tuned a-priori, after a number of experiments. It is not excluded, though, that for some different transfers, a different setting is required. Heuristic rules can be implemented to tune the parameters to always obtain a satisfactory clustering of the solutions.

6.3.2 Integrated Approach

The integrated approach presented in this work is based on a radically new concept, and therefore the research can continue in a number of directions, to better understand and improve the behaviour of the ACO-MGA algorithm. The author believes that there are great possibilities of improving the performances and the range of applications of this technique.

At the time of writing this dissertation, several other test cases have been run (including part of the Laplace mission and interplanetary transfers to outer planets), to assess and compare ACO-MGA on a wider range of transfer problems.

The current most important limitation is related to the search of the lists. In this search method, the lists are interrogated not only to generate the probability distribution, but also for another important reason. The taboo lists are accessed to avoid the exploration of paths that are already known to be unfeasible. Analogously, the feasible list is accessed to find the objective value of a previously explored solution. Both these accesses are performed *instead of* calling the trajectory model. In this sense, the lists can be considered a surrogate model of the trajectory itself. Therefore, a call to the lists saves a trajectory model evaluation. This idea applies not only to the MGA case under consideration, but in general to any problem tackled through the same approach.

With the present implementation, in which the trajectory model is coded in C and compiled, and the lists are coded in MATLAB®, an access to a list can be more expensive than evaluating the model. However, it is expected that a C implementation of the lists would make their access faster than the model itself.

In addition, in the present implementation, feasible and taboo lists are interrogated several times when ants have to build the probability distribution, at each step. It was discovered that the same probability distribution could be generated with a substantial reduction of the number of list interrogations.

Other improvements are also possible. In fact, another reason for ACO-MGA to be computationally slow is the exploration of dead paths. This fact is observed when there are feasible partial solutions that lead to unfeasible ones only. With reference to Fig. 6.1, the partial solution [1, 1] does not appear in the taboo list at level 1, because it is feasible. But if an ant chooses that path, then all its branches are unfeasible. Therefore, the ant cannot explore anymore, as there is no possibility to reach any feasible solution starting with [1, 1]. Computational time has been wasted in two ways: first, it was not needed to explore the leg [1, 1]. Second, the ant had to go through all the possibilities at second leg, for realising that none is feasible. The proposed solution is the equivalent of the back pruning of the incremental approach: i.e. to update the taboo list at previous level (or previous levels) if at a certain point a dead end is found.

In fact the back pruning could be more effective if performed across multiple levels. For example, the feasible partial solution [2, 2] could be added in the taboo list, *even if* not all its branches are unfeasible at the next level. Eventually, all its branches will become unfeasible before reaching the last level.

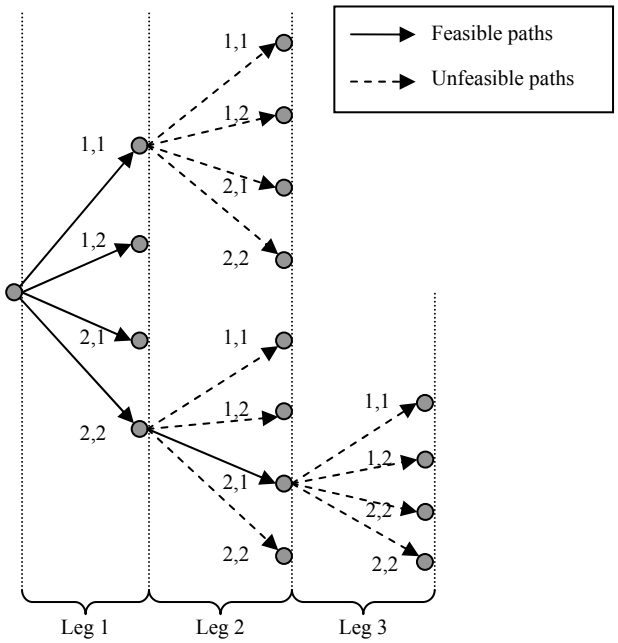


Fig. 6.1. Examples of dead paths in the ideal tree of solutions.

On a longer time-scale, other improvements can be investigated. The current algorithm works only with fixed-length sequences. However, the number of planets involved in the transfer is not always an easy guess. The algorithm can be extended to look up for sequences *up to* a maximum length. This is quite straightforward, as the ants are building the solution leg by leg, and therefore they automatically assess shorter sequences.

At present state, the launch date and launch direction are fixed. Despite, as it was pointed out, a systematic scan can be used along the launch date and the launch direction can be estimated depending on the sequence, the sensitivity on the launch direction can be quite high for some trajectories. Therefore, it could be worth investigating the optimisation of the launch direction inside the trajectory model, together with the excess velocity. This would result in additional cost in the objective function evaluation, but certainly more flexibility for the algorithm.

Concerning the trajectory model used by ACO-MGA, it was explained that currently it is two-dimensional, impeding the use of the algorithm for a number of highly inclined targets. The third dimension could be introduced, avoiding the use of a Lambert solver and more parameters to describe the trajectory. One possible way is to keep the planar problem, but evaluate the cost of the plane change, in terms of Δv , and include this cost in the objective.

Finally, a completely unexplored field of research is the use of the proposed trajectory model, using building blocks, for trajectory design. In this prospective, the number of DSMs, the number and type of swing-bys, and the type of arcs could be found automatically through optimisation. The block model of the trajectory can be exploited in the following way: the optimiser would allocate the blocks meeting

the constraints at interfaces. Each model could implement different models for the same event, each one with different levels of complexity, and number of parameters required. A first step would allocate the blocks and evaluate the trajectory using the simpler models. A second step is called for solutions that are promising at first step, and will use the full models of each block. An integrated approach is also possible, based on nested loops.

Appendix A

IMPLEMENTATION DETAILS

A.1 MATLAB® and C

The incremental pruning is completely coded using MATLAB® (version 2007b was used on the Intel machine and version 2007a on the Sun machine). The trajectory model is almost completely coded in MATLAB®, except for the multi-revolution Lambert routine [96] and the analytical Keplerian propagator [94], which were coded in C and interfaced with the MATLAB® code as MEX-functions. These two routines are the called every time a trajectory has to be evaluated, and they are computationally expensive with respect to the rest of the model, and this is the reason which drove the porting into C. It was found (see Fig. A.1) that the execution of a MEX-function is roughly ten times faster than the corresponding MATLAB® version. The advantage of using FORTRAN with respect to C is negligible.

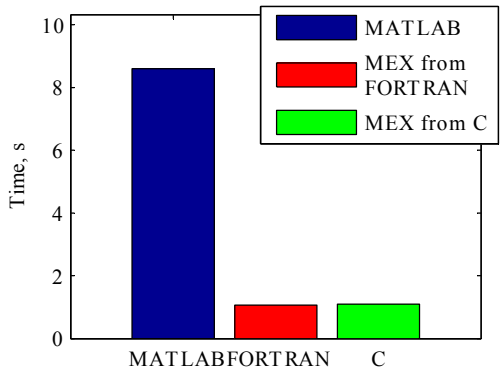


Fig. A.1. Time required for 20000 calls to the analytical Keplerian propagator, using MATLAB® code, MEX-function compiled from FORTRAN and MEX-function compiled from C.

A.2 Solution Refinement

In this subsection, we investigate if some of the minima found by the optimiser actually belong to the same basin of attraction, and the optimiser failed to converge to the same exact point due to high tolerances on termination conditions, or the search space is truly multi-modal, and thus rich in distinct basins of attraction.

To this aim, the solutions of the GCGC transfer case (see Section 4.2.2), which were found through a local optimiser using 10^{-6} as both absolute and relative tolerance, were re-optimised with higher tolerance of 10^{-12}. As visible in Fig. A.2, which is comparing the two sets of solutions, most of the points moved on the left, finding a better value for the minimum, i.e. a better solution in terms of Δv. The number of distinct solutions, instead, is roughly the same. We believe that the search space is dense in local minima, and this is one of the factors which make the MGA-DSM problem so difficult. It was already shown [41] how the introduction of the DSM into ballistic trajectories dramatically increases the number of local minima.

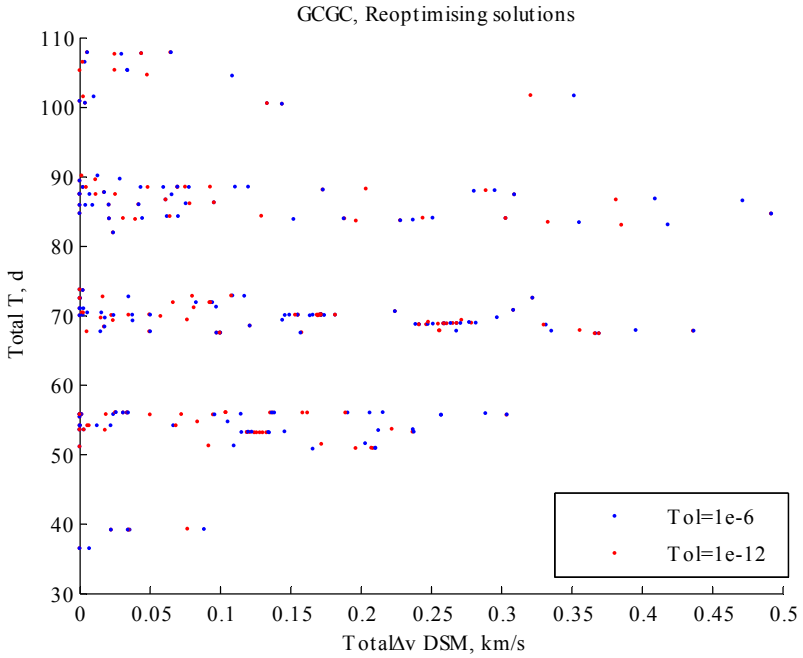


Fig. A.2. GCGC solutions found using a tolerance (both relative and absolute) of 10^{-6} (blue dots) and 10^{-12} (red dots).

Appendix B

MODELLING TRAJECTORIES WITH BUILDING BLOCKS

In this appendix, a general trajectory model paradigm is proposed. The idea is to identify the basic events of a trajectory, and model them by using building blocks. Each block is connected to the following and preceding block through an interface, and the event itself is solved internally to the block.

This approach is very general, and in principle any type of trajectory (as well as other planning problems) can be represented by defining the suitable blocks, interfaces and states. We will present a set of blocks for modelling high-thrust interplanetary trajectories with MGA and multiple DSMs. The actual model within each block will not be explained in detail, for two reasons: the first is that most of the models are either intuitive, or already been covered in the previous two sections. The second is that the functionalities of each individual block do not define the block approach as a whole.

It will be shown that both velocity formulation (Section 2.2) and position formulation (Section 2.3) can be represented as special cases of the block-model.

B.1 Model Description

The main idea in this model is to identify the different events of the interplanetary transfer with *blocks*. Each block models a specific part of the trajectory. It can be an entire leg, or simply a coast arc, a swing-by, or any other event at any complexity level. The block can be seen as a black box, which contains the code for modelling the event, and interacts with other blocks only through an interface, states and parameters. A detailed description of the actual algorithm inside the blocks is beyond the scope of this section, and in fact it is not relevant for the proposed approach. Different blocks can also represent the same type of event, but using different models, or different algorithms: for example a powered swing-by or an unpowered swing-by.

A temporally ordered sequence of blocks defines a specific type of trajectory. Depending on the blocks, the trajectory can have one or more gravity assists, one or more DSMs, and so on. The order of the blocks reflects the temporal order of the events in the trajectory. Blocks in the sequence cannot be overlapped in time.

All the blocks have a property, which is the *duration*: it represents the time that takes to complete the event modelled in the block. An additional attribute of any block is whether its duration is fixed, or it is a free design parameter of the block. As a particular case, a block may have null duration, i.e. the block is instantaneous.

B.1.1 Interfaces

Each block has 2 *interfaces*, to join the following block and the previous block in the temporal sequence. Each interface is intended to match the state of the spacecraft (generally position and velocity) between 2 continuous blocks. The variables on the left hand side of the block refer to the spacecraft state before the event in the block. The variables on the right hand side interface correspond to the spacecraft state after the event.

As an example, let us consider a block which is computing a coast arc through propagation of initial conditions. A block suitable for this task is shown in Fig. B.1. Its left hand side interface has got two variables – position vector and velocity vector – which are inputs to the block. The right hand side interface has got the same two variables, which are outputs. The block will compute the outputs once the inputs are given. The propagation time coincides with the duration of the block.

In the following sections, a more detailed explanation of the elements and features of the block trajectory model will be given.

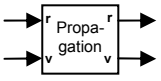


Fig. B.1. The Propagation block: given an initial position and velocity, propagates forward in time. The interface of this block on both sides includes the position vector and the velocity vector.

B.1.2 Interface Types

The *interface type* defines the variables which are used on the interfaces of that type. Therefore, all the interfaces of a particular type have the same variables. Note that the interface type does not specify whether a variable is an input or an output, but only which variables are on the interface.

The interface type is a constraint when building a sequence of blocks: in fact, only blocks with the same type of interface can be adjacent in the sequence. In other words, the interface type must match for two blocks to be connected. Also, there is a constraint related to the input/output nature of each variable: this constraint will be described in a following section.

Three types of interfaces will be used; they are represented in Fig. B.2. For more complicated trajectory models, other interface types could be necessary.

Interface type I_0 has no variables, and it is used on the right side of a block which is a terminator of the trajectory or on the left hand side of block which is a starter for the trajectory.

Interface type I_1 is used in the deep space flight, as in this phase the state of the spacecraft is considered to be fully characterised by its position vector **r** and velocity vector **v**, and each deep space event somehow acts on these vectors.

Interface type I_2 is used when the spacecraft position is supposed to be the same as a planet position. This is the case of a swing-by, for example. Since the heliocentric spacecraft position coincides with the planetary position, and the latter is known through the ephemeris, there is no need to include the position vector in the interface.

Considering again the propagation block of Fig. B.1 as an example, its interfaces are both of type I_1.

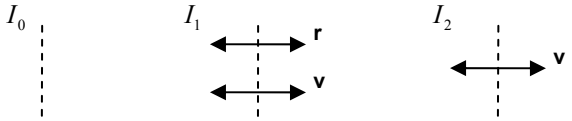


Fig. B.2. The types of interfaces used to model a trajectory.

B.1.3 Interface Variables: Inputs and Outputs. Parameter of Merit

Depending on the block, each variable on the interface can be an *input* or an *output* for that block. As pointed out already, in the propagation block shown in Fig. B.1, both position and velocity are inputs on the left hand side interface, while they are output on the right hand side one. This is directly related to the way the block models the event: in the case of the propagation, the initial position and velocity are required (i.e., before the block), while the result of the propagation is the final position and velocity (i.e. after the block). In the figures here, an input is shown with an arrow going into the block, while for the output the arrow is pointing outwards. Two blocks can be consecutive in a sequence if they have the same interface type and all the inputs on the interface of one block are outputs on the interface of the other block, and vice versa. When these conditions are satisfied for all the couples of continuous blocks, then the sequence is *feasible*. In the representation of the blocks in the figures of this document, the sequence is feasible if the arrows, representing inputs and outputs for each block, have the same direction on the two consecutive interfaces of adjacent blocks, as pictured in Fig. B.3.

Additionally, a *parameter of merit* can be available as an additional output for the block. This is usually the magnitude of the Δv required by the block, but in principle can be any other quantity which will be used to assess the complete trajectory.

For example, let us consider a block which is computing a Lambert arc (Fig. B.4). This block is a way of modelling a coast arc. Its duration is not fixed, and its interfaces may be defined with 2 quantities: the position of the spacecraft and its velocity. Therefore, we will use interface type I_1 on both sides. Since to compute a Lambert arc, the initial and final position shall be given, then we can consider that the position on both the interfaces is an input of the block. In the same way, the result of computing the Lambert arc is the velocity vector at its extremes: so, the velocities are outputs of the block. Since the block models a coast arc, no propelled manoeuvre is involved, and thus there will be no parameter of merit.

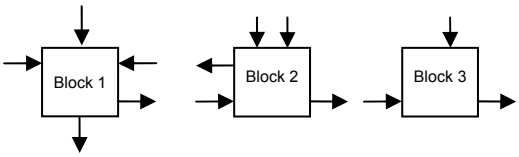


Fig. B.3. An example of a feasible sequence of blocks with different types of interfaces.

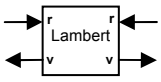


Fig. B.4. The Lambert block, modelling a Lambert arc. Given the initial and final position, computes the initial and final velocity. The time of flight, which is the duration of the block, is computed as a difference of the time state.

B.1.4 Transparency to Variables

Some variables on the interface of some blocks may not be required to evaluate the block, and the event in the block might have no effect on those variables, i.e. the value of those variables is the same before and after the block. In other words, these quantities are neither inputs nor outputs of the block, but they are on the interface. In this case, we say that the block is *transparent* to those interface variables.

Let us consider for example a block which is modelling an (instantaneous) DSM. Since the block models a deep space flight event, the appropriate interface type for it is 1. The block computes the magnitude of the DSM once the velocity before and after the DSM are known: therefore, the velocity is an input on both interfaces of the block. On the other hand, the block does not explicitly need the spacecraft position, and the position does not change before or after the block. The block is transparent for the position, which is neither input nor output on the interfaces. The transparent variable is represented in the figures as a dashed line connecting the two interfaces (Fig. B.5).

When a block is transparent with respect to a variable, for example **r** in the case of the block DSM represented in Fig. B.5, then this variable can be either input or output on the left, but must have opposite attribute on the right, in order to guarantee the feasibility of the sequence, as shown in Fig. B.6.

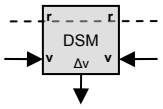


Fig. B.5. The DSM block, computing a deep space manoeuvre. The block is transparent to variable r.

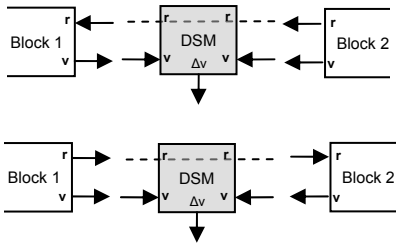


Fig. B.6. Two possible feasible sequences of blocks.

B.1.5 Parameters

A block may have a set of additional input variables, that we will call *parameters*, which are also needed to evaluate the block, but they do not belong to any of the interfaces. The reason for this distinction is that the parameters are not related with the spacecraft state at either side of the block, but they are more connected to the model inside the block itself. Examples of parameters are the launch excess velocity for a launch block, or the radius of pericentre and plane attitude angle for an unpowered swing-by. Note that the parameters could be fixed, or be part of the free design variables of the trajectory.

Each parameter is represented with an arrow on the top of the block, pointing inside the block.

B.1.6 States

The *states* are additional variables defined at all the interfaces of the blocks in the sequence (Fig. B.7). The reasons for making a distinction between states and the interface variables are mainly two. First of all, the states can be computed in advance, for the entire trajectory, without evaluating any block. Second, the states are the same at all the interfaces, regardless the interface type.

A particular state is the time: if the duration of the block is not fixed, then its duration is given by the difference of the time state at its interfaces.

Table B.1 describes briefly the states defined for modelling an MGA trajectory in the following. For more complicated models, additional states may be required.

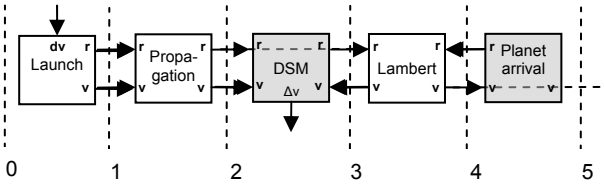


Fig. B.7. Sections at which the states shall be defined.

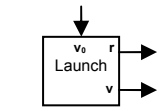
Table B.1. States used to define a trajectory.

State	Description
Time	Epoch of the interface
Previous planet	Planet id of the last encountered planet
Following planet	Planet id of the next planet to be encountered
Current planet	Planet id, if the spacecraft is considered to be at a planet; 0, if the spacecraft is in deep space flight.

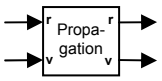
B.1.7 Block Set

In order to model the entire trajectory using blocks, a set of blocks shall be defined, together with their interfaces. The interfaces shall be consistent one another, and reflect the input/output requirements of the block.

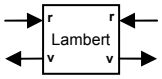
Fig. B.8 shows the basic blocks, defined to reproduce the position formulation and the velocity formulation of the MGA trajectory. A brief description of the block functioning is also provided in the figure.



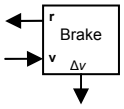
Launch. Fixes position and velocity after the block. It requires the launch excess velocity v_0 as an additional parameter.



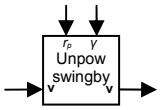
Propagation. Given both position and velocity before the block, it propagates the trajectory for a certain time to obtain position and velocity after the block.



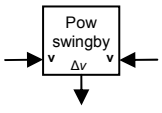
Lambert Arc. It requires the positions before and after the block, and computes the velocities.



Brake. Fixes position (given by the planet ephemeris) before the block, and uses the velocity before the block to compute the brake manoeuvre. The brake Δv is the parameter of merit.



Unpowered swing-by. Given the incoming velocity, it computes the outgoing velocity. It requires 2 additional parameters to determine the swing-by, γ and r_p.



Powered swing-by. It requires both the incoming and the outgoing velocity, and it performs a thrust manoeuvre to match the two legs. The parameter of merit is the Δv needed to perform the powered swing-by.

Fig. B.8. Main blocks for modelling a high thrust MGA trajectory.

In principle, these blocks model all the events used in the velocity formulation and in the position formulation of the trajectory. Note that some blocks have incompatible interfaces: for example, two consecutive Lambert arcs are allowed in the position formulation of the interplanetary leg, but the two corresponding blocks cannot be put next to each other because the input/output configuration is incompatible (Fig. B.9).

Two Lambert arcs should be divided by a DSM. Furthermore, the position of the DSM should be somehow specified. Therefore, the connector blocks shown in Fig. B.10 are introduced. They are represented with a gray box, to underline that they do not contain actual models of any part of the trajectory, but are meant to be connectors for the other blocks.

Using the appropriate connector blocks, it is possible to connect two Lambert arcs as shown in Fig. B.11.

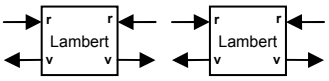
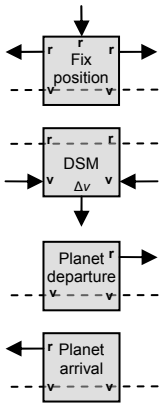


Fig. B.9. Two Lambert arc blocks cannot match because of the inputs/outputs on their interface.



Fix position. Force the spacecraft to pass in a point by fixing the position before and after the block (as given by the external parameter). Transparent to velocity.

Deep space manoeuvre. Computes the change in velocity. Transparent to position.

Planet departure. Used when leaving a planet, it works as an interface between swing-by blocks and deep space flight blocks. Transparent to velocity, fixes the position after the block.

Planet arrival. Used when arriving at planet, it works as an interface between deep space flight blocks and swing-by blocks. Transparent to velocity, fixes the position before the block.

Fig. B.10. Additional blocks for modelling a trajectory.

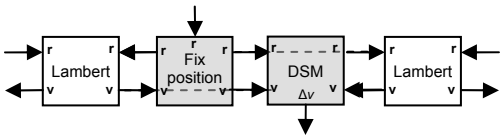


Fig. B.11. Two Lambert arc blocks connected through a Fix position block and a DSM block.

The input-output configuration on the interfaces of the Lambert arc block forces to add additional blocks to match 2 Lambert arcs. A block which fixes the

position of the spacecraft (Fix position) and a block which introduces a discontinuity in the velocity (so a DSM, with related Δv value). This configuration makes sense from a physical point of view. In fact, two matching Lambert arcs require the matching point to be given, and in the same point there must be a discontinuity in the velocity vector (see position formulation). Also note that the order of the two additional blocks used in Fig. B.11 is not important: inverting the two blocks generates the same feasible sequence.

A set of ad-hoc rules was considered, such to insert the appropriate connector blocks in between the main blocks, to make a block sequence feasible, if possible. One example of these rules is the one given above: if two Lambert blocks are next each other, then a DSM and a Fix position blocks must be inserted. Another rule, inspired by the velocity formulation, implies that a Propagation block should be followed by a Lambert block

B.2 Feasibility, Evaluability and Evaluation Order

An ordered sequence of blocks is *feasible* if the interfaces of each couple of continuous blocks are of the same type, and each input parameter on one interface is an output in the other one, and vice versa.

Once a sequence of blocks has been determined, the next step is to compute the trajectory associated to that block sequence, given its solution vector. The solution vector of a trajectory associated with a given sequence of blocks is made by all the parameters of the blocks, and all the necessary time states.

The entire trajectory is evaluated by evaluating all the blocks in the sequence. Evaluating a block means to run the model inside, such that it computes the output interface variables, and the parameter of merit, if available. The feasibility of a given sequence does not guarantee that all its blocks can be evaluated. To be evaluated, a block needs to be provided with all its input interface variables, the states before and after the block, and the parameters. As it was mentioned before, the states can be computed independently of all the blocks, and the parameters of the blocks are in the solution vector: thus they are available. The same cannot be said for the interface variables: their value can only be found by evaluating the other blocks in the sequence.

So there is a constraint which forbids to evaluate the sequence of blocks in temporal order: the fact that a block needs all the input variables on both interfaces to be known, to be evaluated. It follows that there shall be a particular order for evaluating the blocks, such that every time the input interface variables for the next block to be evaluated are available.

In general we can say that a feasible sequence of blocks is *evaluable* if there exists an order in which the blocks can be evaluated, one after the other. This order will be called *evaluation order*, and it is often different than the temporal order of the blocks in the sequence.

Let us consider as an example the block sequence in Fig. B.7. This sequence is temporarily ordered, in the sense that the events represented by each block happen with the same temporal order of the blocks in the sequence. The sequence is clearly feasible.

Its evaluation makes **r** and **v** available at section 1. **r** and **v** at the same section are inputs for the Propagation block, which can in turn be computed, giving **r** and **v** at section 2. The following block, DSM, cannot be evaluated, as the input **v** at its right hand side (section 3) is unknown. Note that, as the DSM block is transparent with respect to **r**, the value of this variable is known also in section 3: it is the same as in section 2. The block Lambert cannot be evaluated either, but it is possible to evaluate Planet arrival, since it has no input interface variables. This completes the Lambert inputs, which in turn completes the DSM inputs. Following these criteria, it results that one possible evaluation order of the sequence is:

Launch → Propagation → Planet arrival → Lambert → DSM

The evaluation order can be represented graphically as in Fig. B.12 considering to have an imaginary horizontal axis of the evaluation order: so the blocks are actually evaluated from left to right in the figure.

An algorithm has been developed to find for any sequence of blocks, if possible, their evaluation order.

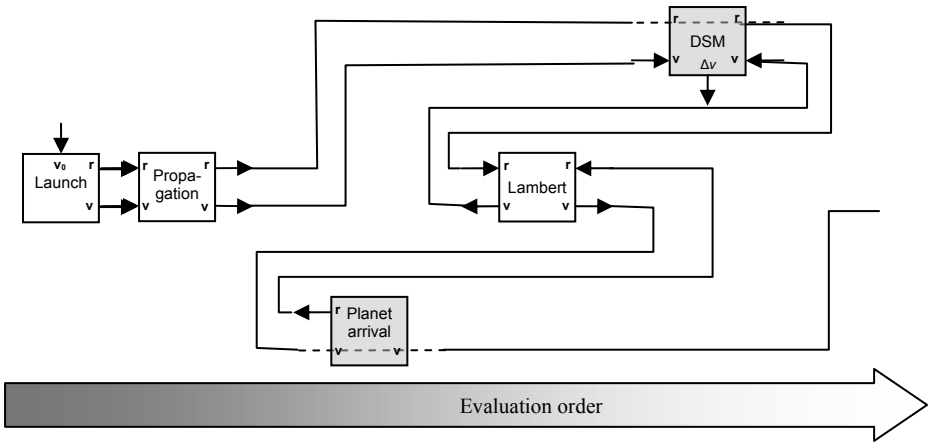


Fig. B.12. Blocks for the sequence in Fig. B.7 positioned along a horizontal axis according to their evaluation order.

B.3 Reproducing Other Models

B.3.1 Velocity Formulation

The velocity formulation model can be reproduced using some of the blocks represented in Fig. B.8 and Fig. B.10. A partial temporal sequence of blocks is

represented in Fig. B.13. The sequence can continue with an arbitrary number of blocks following the same scheme, and depending on the number of legs to consider, and eventually ends with a Brake block. The sequence is evaluable, and its evaluation order is represented in Fig. B.14.

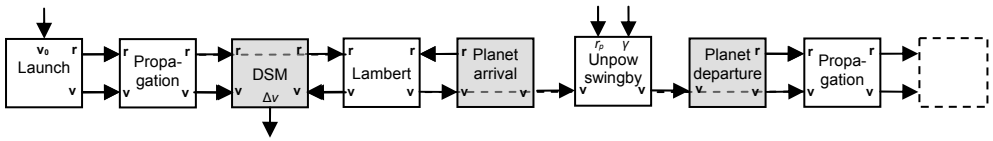


Fig. B.13. Temporal sequence of blocks reproducing a trajectory according to the velocity formulation.

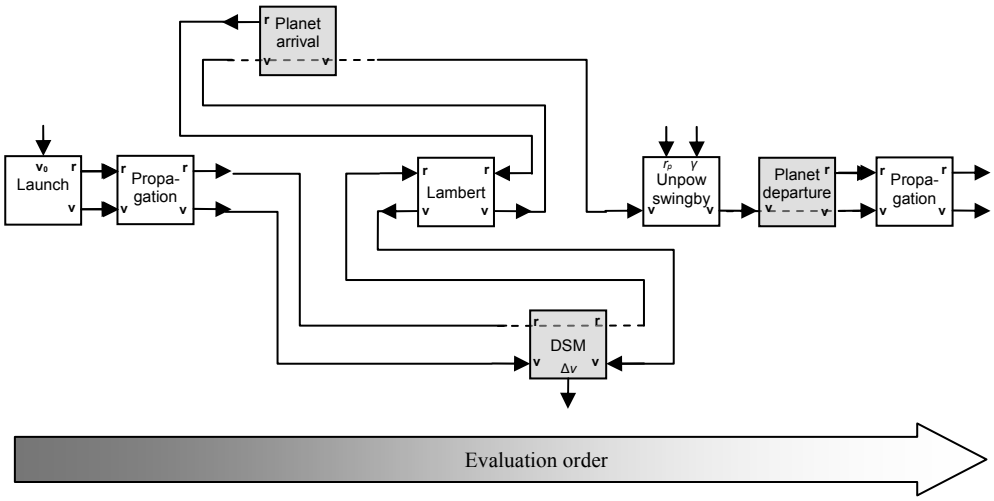


Fig. B.14. Blocks for the sequence in Fig. B.13 positioned according to their evaluation order.

B.3.2 Position Formulation

The position formulation of the trajectory can also be reproduced using the block model. In particular, one deep space flight leg is represented in Fig. B.15. Additional Fix position, Lambert arc and DSM blocks can be added.

The swing-by phase of the position formulation is modelled using the powered swing-by block, which is between the Planet arrival block and the Planet departure blocks (Fig. B.16). The Powered swing-by block can be evaluated when the following and the preceding deep space phases are computed. This reflects the same approach used in the position formulation.

The sequence, regardless the number of DSM and Lambert arc blocks, can be evaluated in the order represented in Fig. B.17.

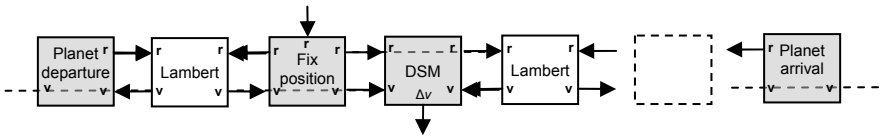


Fig. B.15. Sequence for a deep space flight phase of the position formulation.

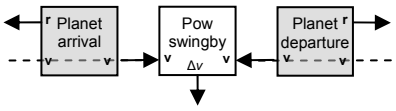


Fig. B.16. Sequence for the swing-by phase according to the position formulation.

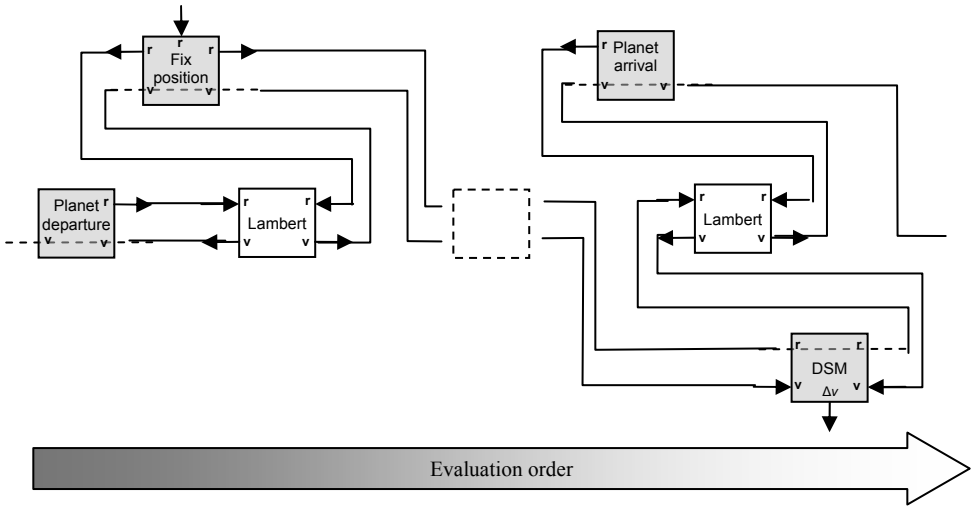


Fig. B.17. Blocks sorted according to the evaluation order for the deep space flight leg of the position formulation.

B.4 Discussion

B.4.1 Incremental Approach

It was shown that every evaluable sequence of blocks can be reordered in the evaluation order. This means that the sequence of blocks can be evaluated incrementally, one after the other. Blocks having a parameter of merit determine the cost of the trajectory up to that point. Moreover, it was explained how to reproduce the position formulation and the velocity formulation with the block approach, and how to generate their evaluation order. Consequently, we can conclude that both the position formulation and the velocity formulation of the trajectory can be solved incrementally, by adding one set of blocks to the trajectory at a time. In Chapter 3, an incremental approach to the MGA trajectory design problem is shown. Even if

the velocity formulation is be used, the same ideas can be applied to the position formulation or in principle to any trajectory generated by an evaluable sequence of blocks.

B.4.2 Different Levels of Accuracy and Detail

An interesting feature of the block-model is that the fidelity of the representation of the trajectory can be improved without changing the block structure. In fact, different blocks can be used to model the same event with different levels of accuracy. As an example, let us consider the Lambert arc block. Solving the Lambert problem is known to be computationally expensive with respect, for example, to a propagation. So a very quick (but less precise) Lambert solver can be used in the first place, for a fast assessment of the trajectory. Then, in a second time, a more accurate Lambert solution can be plugged into the same block without modifying the overall structure.

The block model allows also for a hierarchical construction of the trajectory. A first coarse solution can be built with few blocks modelling large sections of the trajectory (possibly using reduced or simplified models). A second solution is then computed using sub-blocks of the coarse solution modelling smaller events with higher accuracy.

For example, it could be possible to define a block which is coding an entire interplanetary leg, providing a rough estimation of the necessary Δv, possibly using some heuristic rules. In a second time, a more accurate trajectory can be computed by un-plugging the block, and inserting a number of Lambert arcs, low-thrust arcs, propagation arcs, and DSMs.

B.4.3 Automatic Trajectory Planning

A further interesting feature offered by the block model is the automatic planning of complex trajectories. By *planning* we mean the process of finding the best temporal sequence of blocks to achieve a certain mission goal.

A sequence of blocks can be easily coded in a vector of discrete variables, in which the value of each entry represents a block in the sequence. Then, an optimisation algorithm for discrete problems can be used to find an optimal block sequence. This concept is expanded in Chapter 5, where a particular planning and scheduling approach will be presented.

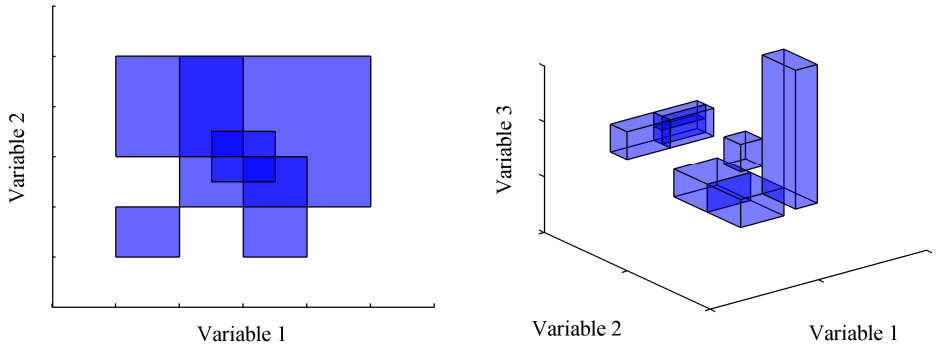
Appendix C

AFFINE TRANSFORMATION

While the pruning process reduces the volume of the search space, the resulting residual space is not necessarily box-constrained. If one of the clustering and boxing techniques is applied (for example those in Section 3.4.2), then the residual space can be represented as a set of boxes, defined on each level $D_{L,i}$ or on the whole domain up to the level D_i. Nevertheless, the resulting space remains in general disconnected. Fig. C.1 shows examples of disconnected and/or overlapped boxes for a two-dimensional and a three-dimensional case.

An additional problem is that the number of boxes could grow exponentially, and so would the number of subsequent optimisations, if an optimisation for each box is considered.

On the other hand, the pruning is advantageous only if the search, at the following level, is limited on the non-pruned part of the space, and no effort is made to search on the residual part. Therefore, it is essential to search only on the non pruned part of the domain.



(a) Two dimensions (b) Three dimensions

Fig. C.1. Two examples of domains defined as a set of boxes. The boxes can be partially or completely overlapped.

C.1 Affine Transformation

A possible solution is to collect all the boxes at each level, such to pack them in a way that they fill a *transformed* box-constrained search space. The search is then performed on this transformed space, rather than on each box individually.

In order to pack all the (disconnected) boxes generated at each level *i*, a space transformation is applied that maps all the boxes into a unit hypercube made of connected boxes. If level *i* is under pruning, then a transformed space is generated for each level $k \le i$.

The dimensionality of the transformed space is the same as the one of the original level space, so is the number of partitions in the unit hypercube. Each box in the real space has a corresponding partition in the transformed space, and a linear transformation allows mapping a point $\tilde{\mathbf{x}}_{L,i}$ inside a box in the transformed space into a point $\mathbf{x}_{L,i}$ in the corresponding box in the real space:

$$x_{L,i,j} = \frac{\left(b_{u,L,i,j} - b_{l,L,i,j}\right)}{\left(\tilde{b}_{u,L,i,j} - \tilde{b}_{l,L,i,j}\right)}\left(\tilde{x}_{L,i,j} - \tilde{b}_{l,L,i,j}\right) + b_{u,L,i,j} \tag{C.1}$$

for each dimension *j* of the level *i* under consideration. $\tilde{\mathbf{b}}_{u,L,i}, \tilde{\mathbf{b}}_{l,L,i}$ are the upper and the lower bounds of the box in the unit hypercube which contains $\tilde{\mathbf{x}}_{L,i}$, and $\mathbf{b}_{u,L,i}, \mathbf{b}_{l,L,i}$ are the bounds of the corresponding box in the real space.

Using this transformation of the search space, it is possible to run the search for feasible solutions on the unit hypercube, as schematised in Fig. C.2. The affine transformation is bi-univocal, thus it allows obtaining the point in the real space given a point in the unit hypercube, and evaluating the objective function in that point. In such a way, the optimisation problem becomes box constrained, and a generic optimiser can be used to search the space.

It is also to be noted that the affine space contains all and only the boxes on which the function is defined: thus, only those parts of the search space which have not been pruned out are included in the search.

In general, there exist infinite ways to partition the unit hypercube in a given number of boxes: some examples are shown in Fig. C.3 for a two-dimensional space. Each of those may have different properties and drawbacks. In this work, two methods have been studied and used, each one trying to pursue a certain property.

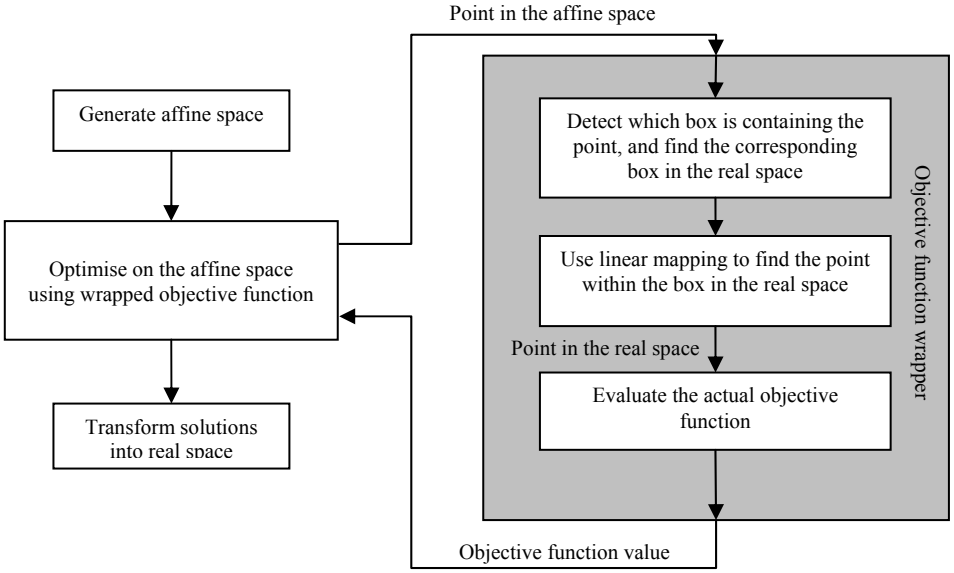


Fig. C.2. On the left, steps to search on the affine space. On the right, the wrapped objective function used to search on the affine space.

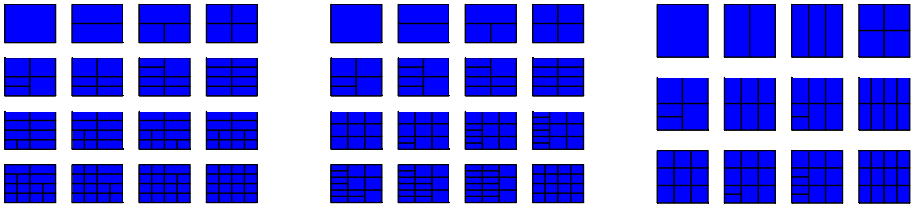


Fig. C.3. Three different ways to partition the affine space for a required number of boxes from 1 to 16.

C.1.1 Method 1

The first method aims at partitioning the unit hyper-cube in a way that most of the boxes have the same shape (and thus the same volume). The idea is that, if the boxes have the same volume and the same edges, the probability of sampling each box of the space is uniform. Being $\left|D_{L,i}\right|$ the number of dimensions of the generic level i, and q_i the number of boxes on that level, it is possible to meet these two requirements together only if:

$$p = \sqrt[\left|D_{L,i}\right|]{q_i}$$

is an integer. p is the number of intervals to consider on each dimension to partition the affine space into q_i hyper-cubes. In all the other cases, it is still possible to have boxes of equal size (for example cutting the hypercube along only one coordinate), but the number of subdivisions per coordinate would be uneven. This means that

the objective function will be more sensitive to any change of that coordinate along with the cuts have been made. This situation is undesirable for any optimiser.

Algorithm C.1 was developed to try to keep the boxes similar one another in dimensions, and at the same time, with similar edge length. The pseudo-code is processing each dimension at a time, and determines p_j, which is the number of subdivisions along the j^{th} dimension. q_{left} is a variable to keep track of the number of remaining boxes to generate, once a dimension has been processed. At the end of the algorithm, if $q_{left} \neq 0$, then the regular subdivisions could not generate all the required boxes, so further boxes (which will be different in size) are generated by halving existing boxes.

Fig. C.4 shows the partitioned unit hypercube, for a number of partitions from 1 to 12: (a) is the case of a two-dimensional space, while (b) is for a three-dimensional space. The algorithm works in the same way for an arbitrary number of dimensions. This method provides regular boxes in most of the cases: uneven partitioning is noticeable for 3, 6, 8, 12, 15 boxes. However, the generation of boxes in the real space is arbitrary therefore it is always possible to generate the optimal number of boxes.

It is noteworthy that, in order to partition the unit hypercube with this algorithm, no information about the size of the boxes in the real space is required.

Algorithm C.1. Generation of the affine space according to method 1.

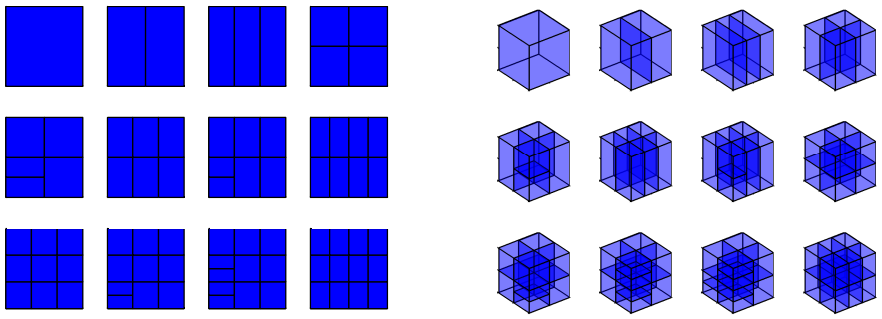
1: $q_L \leftarrow q$

2: **For** $j = \left|D_{L,i}\right|, \left|D_{L,i}\right| - 1, \ldots, 1$

3: $p_j \leftarrow \text{floor}\left(\sqrt[j]{q_{left}}\right)$

4: $q_{left} \leftarrow \dfrac{q_{left}}{p_j}$

5: **End For**



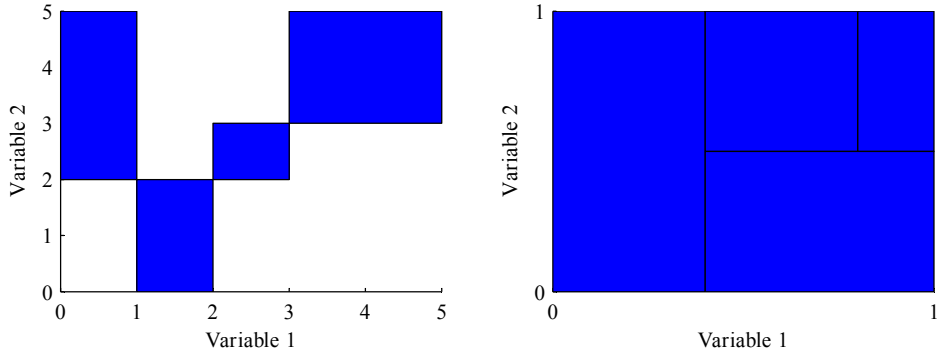
(a) 2D case (b) 3D case

Fig. C.4. Partitioning of the unit hyper-cube for different numbers of partitions (from 1 to 12), and in the 2D case (a) and in the 3D case (b), using method 1.

C.1.2 Method 2

The second method for partitioning the unit hypercube aims at preserving the mutual proportions of the volume among the boxes. In this case, in addition to the number of boxes, it is necessary to compute the volume of each box in the real space, relative to the total volume of all the boxes. Called the relative volume V_i for a generic box i, this must be actual volume of the corresponding box in the unit hypercube, as the total volume of the unit hypercube is unitary by definition. After having sorted the boxes by their volume, in decreasing order, an iterative procedure is initiated. Starting with the first box, one coordinate of the affine space is cut such that the volume of one the resulting two boxes is V_1. Then the second box is considered in the sorted list and the same procedure is applied to the other box of the affine space. The iterative bisection process continues until all the boxes have been processed. At each iteration, the cut is performed cyclically on each coordinate.

The choice of maintaining the volume ratio is made to preserve the sampling probability of the original space $\bar{D}_{L,i}$, when a uniform sampling is performed. Note that this algorithm is partitioning the unit hypercube maintaining the mutual proportions among the volumes, but not among the length of the edges. A two-dimension example of the space partitioning is shown in Fig. C.5.



(a) Real space (b) Affine space

Fig. C.5. An example of partitioning the unit hyper-cube using method 2: (a) the real space; (b) the affine space, partitioned accordingly.

C.2 Discussion

The value of the pruning function f_i at level i is then evaluated by sampling the affine space for levels $1,...,i$ and then mapping the sampled points into the real space.

The creation of the affine space (and the search on it) is a solution to the problem of optimising a function defined on a set of boxes on various levels,

avoiding the exponential growth of the number of optimisations with the number of levels, if a different search for box is considered.

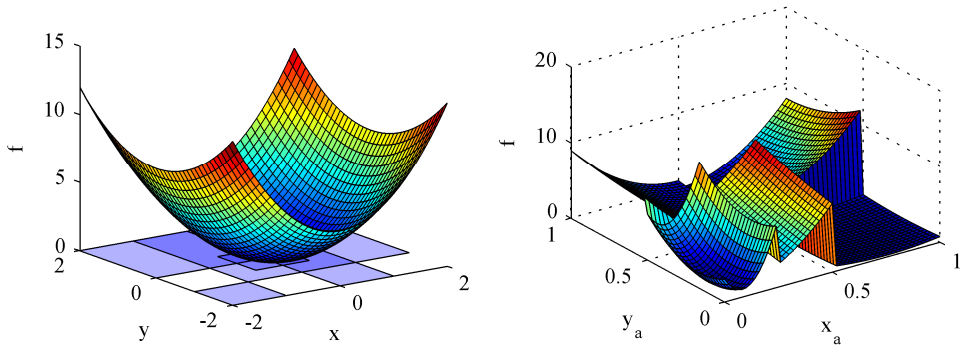
The alternative to the space transformation is to run several searches on each box independently, each one of those being box constrained, and then merge the results. If there is only one level, the two methods are quite similar. If there are q boxes, it is possible to run either q optimisations, each one using a different box as boundary, or only one on the affine space. Since the affine space maps to all the boxes, a better search is needed in order to obtain solutions of the same quality as before. If we assume that the optimiser is distributing some agents in the search space, with a given density with respect to the characteristics of the function, the total number of agents used in the q optimisations on the boxes separately should be similar to the number of agents used to optimise the affine space once.

Now let consider that there are two levels, with dimensionality n_1 and n_2, with q_1 and q_2 boxes respectively, defined on their respective levels. If we want to search on each separate box-constrained region (i.e. without using the affine transformation), we need to run one search for each possible pair of boxes from level 1 and 2. This means that we need $q_1 \cdot q_2$ optimisations, each one on a space with $n_1 + n_2$ dimensions. For a general problem with n_{legs} levels, the optimisations needed are $\prod_{i=1}^{n_{legs}} q_i$.

Using the affine transformation for each level, the affine space will have $n_1 + n_2$ dimensions, and only one optimisation is needed. Of course, even if the dimensionality of the optimisation is the same as in the case of optimising one set of boxes, many more agents are needed here in order to guarantee the same quality of search. This is due to the fact that the affine space is built by joining all the boxes for each level. In spite of this, there might be an advantage by using the affine space. In fact, in this case, the optimiser is free to move in the entire affine space, which is the same as moving in *all* the boxes. The optimiser does not see the box partitioning of the affine space. So the optimiser can choose the best combinations of boxes for each level just by moving in the affine space.

However, it should be noted that the objective function seen from the affine space is discontinuous even if it is continuous in the real space (Fig. C.6). This is due to the way the space has been created: it is composed by disconnected and/or overlapped boxes in the real space. This makes the use of gradient-based techniques in particular quite tricky.

The discontinuities of the objective function depend on how the boxes are connected in the affine space. However, if the boxes are disconnected or overlapped in the real space, the objective function in the affine space results to be always discontinuous. In addition the number of local minima in the affine space may be greater than in the real space. Although this might seem a pitfall, it should be noted that in practice a good deal of the minima in the affine space are replica of individual minima in the real space. As a consequence there is an increased probability to find a good solution.



(a) Real space (b) Affine space

Fig. C.6. In (a), a paraboloid defined on a set of boxes, and in (b), the corresponding function in the affine space.

The idea of generating the boxes by merging together all the regions with feasible points within one single enveloping box (by using method 3 or 4) has the advantage that the local structures of the objective function are preserved through the affine transformation in the transformed space, ensuring that every box tries to identify a particular region of interest, and no regions of interest are split among different boxes.

C.3 Test Cases

The following subsections present two additional test cases using the affine transformation. They exploit boxing method 3 (refer to Section 3.4.2).

C.3.1 Sequence EEM

This test case is very similar to the one presented in Section 3.6.2 (to which we refer for a complete description of the problem), but the objective function is modified to include a final orbit insertion manoeuvre[1]. Therefore, the function f is the sum of the Δv of the two DSMs, plus the Δv_f needed to inject the spacecraft into an ideal operative orbit around Mars with 3950 km of pericentre radius and 0.98 of eccentricity [104]. The bounds are the same as those presented in Table 3.10.

The incremental pruning was run with 30 randomly distributed starting points for level 1 and 20 for level 2. The threshold for $f_1 = \Delta v_1$ at level 1 was set to 0.5 km/s, and the Multi-Start optimiser was stopped as soon as a solution reaches this

[1] This test case was run during an ARIADNA study in collaboration with ESA-ACT and the University of Reading. The objective function was modified on explicit request by ESA for sake of comparison with other approaches.

value. Due to the different method for generating boxes, the size of the edges at level 1 for each variable was set to the values represented in Table C.1.

The result of the pruning of level 1 is shown in Fig. C.7 and Fig. C.8, which represent the projection of the boxes along variables of level 1. The red dots represent the solutions found by the Multi-Start optimisation at level 1. All the search space which is not included in one of the boxes is pruned out, and not considered during the search at the following level.

Table C.1. Box size for the EEM test case.

Variable	Box edge
t_0 , d	Whole domain
$\overline{\delta}$, rad	0.1429
$\overline{\theta}$, rad	0.1429
α_1	0.2967
T_1 , d	95 d

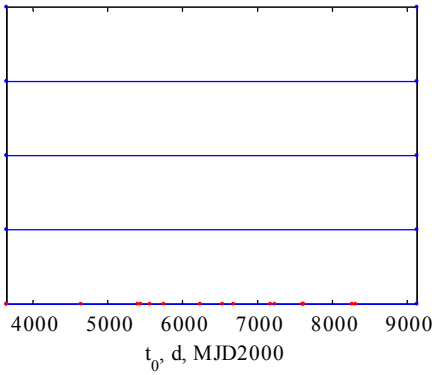
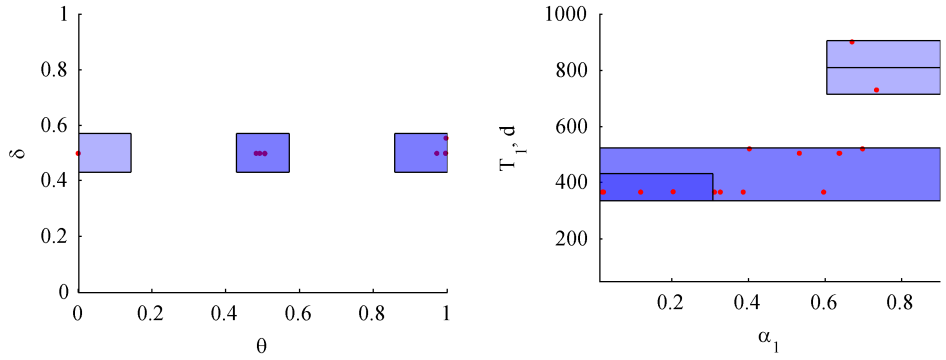


Fig. C.7. Projection on t_0 of boxes and solutions after pruning level 1.



(a) Projection on δ, θ (b) Projection on T_1, α_1

Fig. C.8. Boxes and solutions after pruning level 1.

The fact that t_0 is not relevant is clear from Fig. C.7, as the solutions are spread along the whole time span. Instead, the pruning process reduces the search space along the other variables considerably, in particular $\overline{\theta}$, $\overline{\delta}$ and T_1. Fig. C.8 (a) reveals that all the solutions have the non-dimensional declination $\overline{\delta}$ at 0.5, which means that the launch must be in the Earth orbit plane, and non-dimensional declination $\overline{\theta}$ equal to 0, 0.5, or 1: these values correspond to a launch excess velocity aligned with Earth orbital velocity (with the same or the opposite direction). Considering the time of flight T_1, there are 4 classes of solutions, around 365 d, 510 d, 720 d and 900 d. These local minima have been clustered in 3 boxes, as seen in Fig. C.8 (b).

The following step of the incremental algorithm is the process of level 2. Since level 2 is the last one in this problem, its pruning is not necessary. All the solutions found by the Multi-Start optimisation are sorted and the best one is considered the best global minimum. The search for the solutions at level 2 takes advantage of the pruning at level 1, and exploits the space transformation.

The smallest Δv found by the incremental approach, averaged on the 20 runs, is shown in Table C.2, together with the same value obtained by running Differential Evolution, Particle Swarm Optimization and the Multi-Start on the complete problem all-at-once. The number of objective function evaluations is also shown, as a parameter of the computational power required to obtain a certain objective value, and thus as an index of the performance of the optimiser. For the incremental approach, the number of function evaluations for each level is shown. The standard deviation of the best-found objective value on the 20 runs is also shown.

The result is that the incremental algorithm finds solutions with a lower Δv than DE, PSO and the Multi-Start optimisation, with about 1/10 of the function evaluations.

The trajectory corresponding to the best solution found by the incremental approach is represented in Fig. C.9.

Table C.2. EEM results for 4 different approaches, values computed on 20 runs.

Optimiser	Average no. of function evaluations	Best Δv [km/s]	
		Average	Standard deviation
DE all-at-once	200070	1.591	0.136
PSO all-at-once	200000	1.556	0.238
Multi-Start all-at-once	210217	1.268	0.137
Incremental	6097, 18519	1.171	0.081

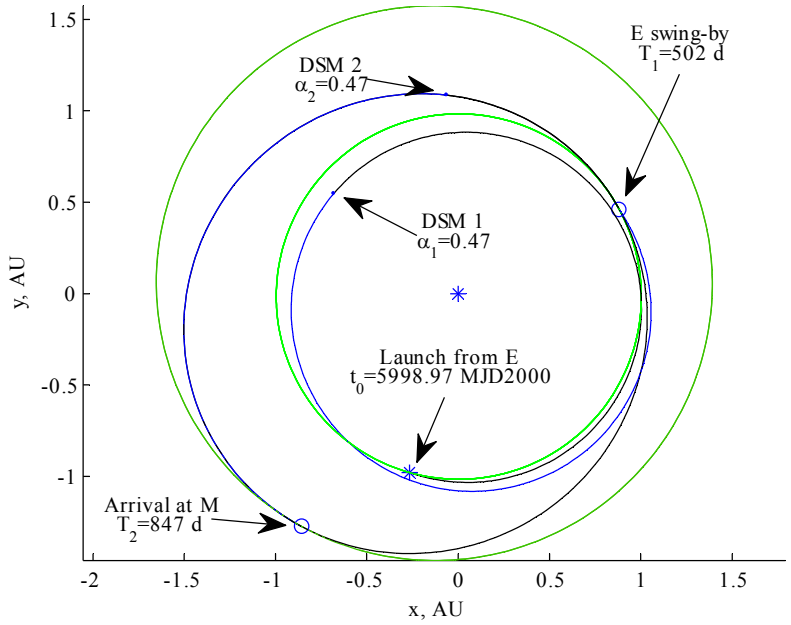


Fig. C.9. Projection on the ecliptic plane of the best solution found by the incremental algorithm. The total Δv is 1.08 km/s, including the final orbit insertion manoeuvre.

C.3.2 Sequence EEVVMe

This sequence to reach Mercury includes three swing-bys, 2 of which are resonant. It is the same sequence chosen for the MESSENGER mission [25].

The launch excess velocity was fixed to 1.5 km/s, and no orbit insertion manoeuvre was considered at Mercury, because other resonant swing-bys may be added to further slow down the spacecraft. The objective function is then the sum of the DSMs in each leg. The bounds for this problem are shown in Table C.3.

The 4th leg was required to perform 6 revolutions around the Sun. To this aim, the bounds on α_4 were restricted such that the propagated part of the leg can perform at least 3 complete revolutions, while the subsequent Lambert problem is solved searching for a 2-complete-revolution solution.

For the incremental approach, 100, 100, 100, 200 starting points for levels 1 to 4 respectively were used. The size of the boxes for level 1 was set to a fraction of the span of the space (apart from t_0), as shown in Table C.4. At the pruning of level 2, the back pruning was used, and the boxes were re-generated also at level 1, with the size associated to variable t_0 reduced to 1/10 of the original size. The reason is that the E-V leg introduces some constraints on the phasing of the Earth-Venus system, and this reduces dramatically the range of the possible launch dates, in order to have a low Δv. For the variables of the levels 2 to 4, the size of the boxes was kept fixed, as in Table C.4.

Table C.3. Bounds for the EEVVMe test case.

Variable	Lower bound	Upper bound	Level
t_0, d, MJD2000	4500	5500	
$\overline{\theta}$	0	1	
$\overline{\delta}$	0	1	1
α_1	0.2	0.9	
T_1, d	350	600	
γ_1, rad	$-\pi$	π	
$r_{p,1}$, planet radii	1	5	2
α_2	0.01	0.99	
T_2, d	300	450	
γ_2, rad	$-\pi$	π	
$r_{p,2}$, planet radii	1	5	3
α_3	0.01	0.99	
T_3, d	150	300	
γ_3, rad	$-\pi$	π	
$r_{p,3}$, planet radii	1	5	4
α_4	0.595	0.733	
T_4, d	750	850	

Table C.4. Box size for the EEVVMe test case.

Level under pruning	Box edges at level 1	Box edges at level 2	Box edges at level 3	Box edges at level 4
1	t_0 : whole $\overline{\delta}$: 0.2 $\overline{\theta}$: 0.2 α_1 : 0.2333 T_1 : 50 d			
2…4	t_0 : 100 d $\overline{\delta}$: 0.2 $\overline{\theta}$: 0.2 α_1 : 0.2333 T_1 : 50 d	γ_1 : 1.25 rad $r_{p,1}$: 1.33 α_2 : 0.29 T_2 : 30 d	γ_2 : 1.25 rad $r_{p,2}$: 1.33 α_3 : 0.29 T_3 : 30 d	γ_3 : 1.25 rad $r_{p,3}$: 1.33 α_4 : 0.046 T_4 : 20 d

The objective function in Eq. (3.17) was chosen for searching the solutions on level 1 and 3 of the incremental approach. These levels correspond to the resonant swing-by legs E-E and V-V respectively. For levels 2 and 4, the sum of the Δv was

chosen. Local minima above 1, 1.1, 1.2 km/s were discarded at levels 1, 2, 3 respectively.

As in the EEM case, the pruning of level 1 does not identify any particular launch window, even if some periodicity is visible due to the eccentricity of the Earth orbit (Fig. C.10). As seen in Fig. C.11, the incremental algorithm clearly identifies an in-plane ($\bar{\delta} = 0.5$), tangential ($\bar{\theta} = 0, 0.5, 1$) launch direction; Furthermore, there are three classes of solutions with 3 possible times of flight T_1, clustered into 2 sets of boxes. The solutions are spread in a wide range on α_1.

The search at level 2 reveals that the solutions are no more spread along t_0 (Fig. C.12): thus, the reduced size of the boxes allows identifying a few launch windows, between 4900 and 5200 MJD2000 in particular. This result was expected and justifies the choice for a smaller box size along t_0 after level 1. In Fig. C.13 (a) and b, it is noticeable that the time of flight T_2 for the E-V leg should be around 430 d. The projection of the boxes along the axes of the Earth swing-by shows that the ideal Earth swing-by angle γ_1 is around 0. No pruning is done on $r_{p,1}$, as the solutions are spread in the whole span.

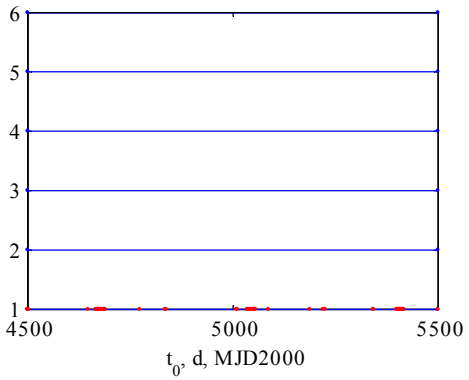
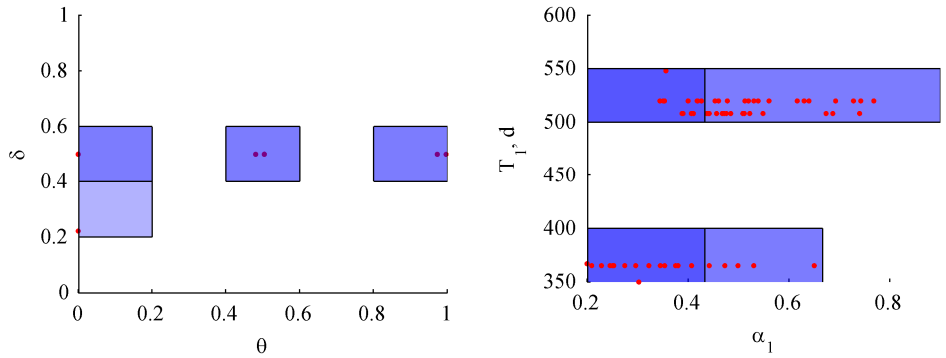


Fig. C.10. Projection on t_0 of boxes and solutions after pruning level 1.



(a) Projection on θ, δ (b) Projection on α_1, T_1

Fig. C.11. Boxes and solutions after pruning level 1.

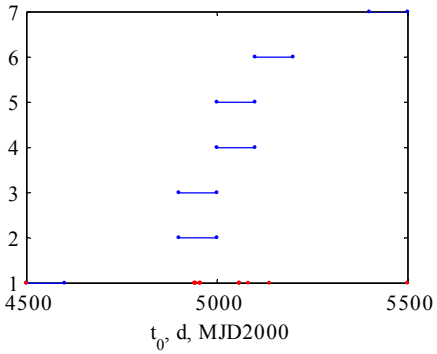
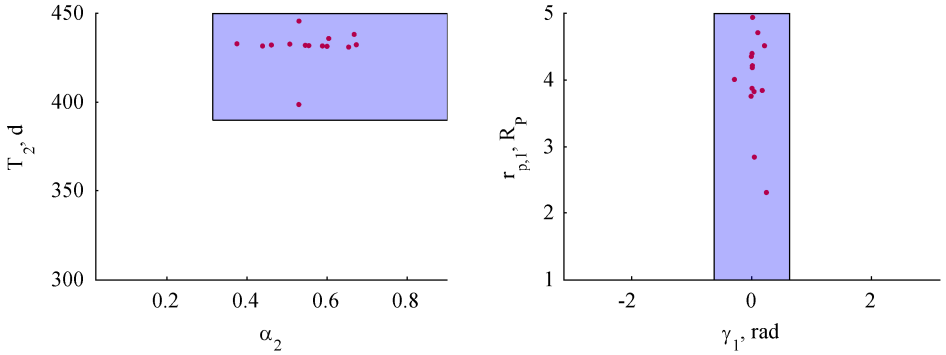


Fig. C.12. Projection on t_0 of boxes and solutions after pruning level 2.



(a) Projection on α_2, T_2 (b) Projection on γ_1, $r_{p,1}$

Fig. C.13. Boxes and solutions after pruning level 2.

Table C.5. EEVVMe results for 4 different approaches, values computed on 20 runs.

Optimiser	Avg. no. obj. fun. eval.	Time for obj. fun. eval. [s]	Best Δv [km/s] Avg.	Std. dev.
DE all-at-once	400010	5842	8.456	0.444
PSO all-at-once	460000	6900	6.094	0.920
Multi-Start all-at-once	427499	6412	4.599	0.865
Incremental	24397, 96674, 184340, 154754	3625	3.89	0.739

The incremental approach proceeds in the same way up to level 4. At this point, the near-global optima are found. Table C.5 shows the comparison of the incremental approach with the two all-at-once approaches, in terms of objective function and number of function evaluations.

For this test case, the incremental algorithm outperforms all the other methods, using about the same number of function evaluations. Nevertheless, it has to be considered that all the all-at-once approaches evaluate the objective function for the complete problem every time, while the incremental is evaluating that function only at level 4. The partial objective functions at lower levels are cheaper to compute, as they include less legs and zero-revolution Lambert problems, which are quicker to solve than the multi-revolution one at leg 4. Times for one function evaluation on an Intel Pentium 4 3 GHz are reported in Table C.6. The result is that the total time spent in evaluating the objective function is far lower for the incremental approach than for the others. At the same time the incremental approach was able to identify better trajectories.

Fig. C.13 plots the projection of the best trajectory found by the incremental algorithm during one of the 20 runs. The total Δv is 4.55 km/s with a relative velocity at Mercury of 8.2 km/s. Note that the reason why the relative velocity at Mercury is so high is that it was not included in any pruning criterion or in the objective function of the whole problem.

Table C.6. Average time to evaluate the partial objective functions, for each level, in seconds.

Level	1	2	3	4
Time, s	$1.8\cdot10^{-3}$	$3.5\cdot10^{-3}$	$5.0\cdot10^{-3}$	$1.5\cdot10^{-2}$

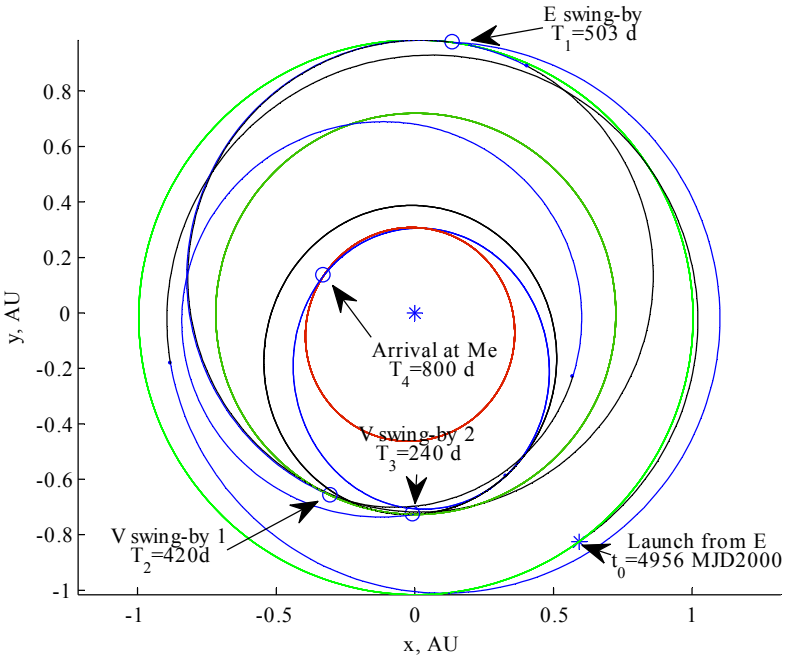


Fig. C.14. Projection on the ecliptic plane of the best solution found by the incremental algorithm.

Appendix D

TESTING PROCEDURE FOR GLOBAL OPTIMISATION ALGORITHMS

If we call A a generic solution algorithm and p a generic problem, we can define the procedure in Algorithm D.1.

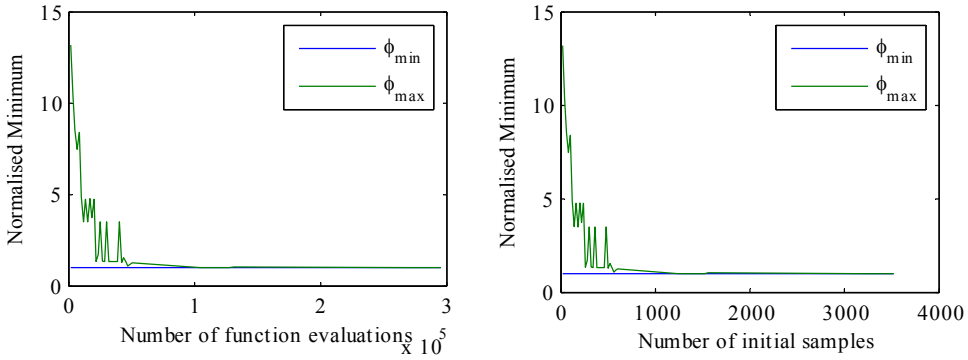
Now and in the following we say that an algorithm A is globally convergent, when for a number of function evaluations N that goes to infinity the two functions ϕ_{min} and ϕ_{max} converge to the same value, which is the global minimum value denoted as f_{global}. An algorithm A is simply convergent, instead, if for N that goes to infinity the two functions ϕ_{min} and ϕ_{max} converge to the same value, which is not necessarily a global or a local minimum for f.

If we fix a tolerance value tol_f, we could consider the following random variable as a possible quality measure of a globally convergent algorithm:

$$N^* = \min\left\{N : \phi_{max}(N') - f_{global} \le tol_f, \forall N' \ge N\right\}. \tag{D.1}$$

Algorithm D.1. Convergence test.

1: Set the max number of function evaluations for A equal to N
2: Apply A to p for n times
3: **For** $i \in [1,...,n]$, **Do**
4: $\quad \phi(N,i) = \min f\left(A(N), p, i\right)$
5: **End For**
6: Compute $\phi_{min}(N) = \min_{i\in[1,...,n]} \phi(N,i)$; $\phi_{max}(N) = \max_{i\in[1,...,n]} \phi(N,i)$



(a) Number of function evaluations (b) Number of initial samples

Fig. D.1. Convergence profile for a bi-impulsive Earth-Apophis transfer. (a) Convergence as a function of the number of function evaluations; (b) Convergence as a function of the number of initial samples for a Multi-Start algorithm.

The larger (the expected value of) N^* is, the slower is the global convergence of *A*. Fig. D.1 (a) and (b) show the convergence profile for the bi-impulsive problem obtained with 50 repeated independent runs of a Multi-Start algorithm: a number of samples were generated in the solution space with a Latin hypercube sampling procedure and a local optimisation was run from each sample. Slightly more than 1000 initial samples are required to have a 100% convergence to the global minimum. However, the procedure in Algorithm D.1 can be impractical since, although finite, the number N^* could be very large. In practice, what we would like is not to choose *N* large enough so that a success is always guaranteed, but rather, for a fixed *N* value, we would like to maximize the probability of hitting a global minimiser. Now, let us define the following quantities:

$$\delta_f(\mathbf{x}) = \left| f_{global} - f(\mathbf{x}) \right|;\ \delta_x(\mathbf{x}) = \left\| \mathbf{x}_{global} - \mathbf{x} \right\|. \tag{D.2}$$

In case there is more than one global minimum point, $\delta_x(\mathbf{x})$ denotes the minimum distance between $\mathbf{x}$ and all global minima. Moreover, in case the global minimum point $\mathbf{x}_{global}$ is not known, we can substitute it with the best known point $\mathbf{x}_{best}$. We can now define a new procedure, summarised in Algorithm D.2.

A key point is setting properly the value of *n*. In fact, a value of *n* too small would correspond to an insufficient number of samples to have proper statistics. The number *n* is problem dependent and is related to the complexity of the problem and to the heuristics implemented in the solution algorithm. A proper value for *n* should give a little or null fluctuations on the value of j_s/n, i.e. by increasing *n* the value of j_s/n should remain constant or should have a small variation. Note that the values of the tolerance parameter tol_f and tol_x depend on the problem at hand. Algorithm D.2 is applicable to general problems either presenting a single solution

with value function f_{global} (or f_{best}) or presenting multiple solutions with value f_{global} (or f_{best}). On the other hand, in the following we are not interested in distinguishing between solutions with equal *f* and different **x** therefore we will use a reduced version of Algorithm D.2 in which the condition $\delta_x(\mathbf{x}) \le tol_x$ is not considered.

Finally, we remark that the two procedures described in Algorithm D.1 and Algorithm D.2 only consider the computational cost to evaluate *f* but not the intrinsic computational cost of *A*. The intrinsic cost of *A* is related to its complexity and to the number of pieces of information *A* is handling. For instance, for a simple grid search such intrinsic cost is represented by the cost of sweeping through all the *N* points on the grid at which the objective function is evaluated. The intrinsic cost varies from algorithm to algorithm, but here we are assuming that the computational effort of the algorithms is dominated by function evaluations and, therefore, we do not take intrinsic costs into account. Note that if the algorithm *A* is deterministic, then we can set $n=1$. Indeed, each time *A* is applied to *p*, it always returns the same value. Then, for deterministic algorithms, given a value *N*, a reasonable performance index is simply $J_d(N)=\phi(N,1)$, i.e. the best value returned by the algorithm. Instead, for stochastic based algorithms different performance indexes can be defined. Such indexes are computed by running an algorithm over a problem a sufficiently high number *n* of times.

Algorithm D.2. Convergence to the global optimum.

1: Set the max number of function evaluations for *A* equal to *N*
2: Apply *A* to *p* for *n* times
3: Set $j_s = 0$
4: **For** $i \in [1,...,n]$, **Do**
5: $\quad \phi(N,i) = \min f\left(A(N), p, i\right)$
6: $\quad \mathbf{x} = \arg\phi(n,i)$
7: $\quad$ Compute $\delta_f(\mathbf{x})$ and $\delta_x(\mathbf{x})$
8: $\quad$ **If** $\delta_f < tol_f \wedge \delta_x < tol_x$ **Then**
9: $\quad\quad j_s = j_s + 1$
10: $\quad$ **End If**
11: **End For**

Commonly used indexes are the best, the mean and the variance of all the results returned by the *n* runs, or the probability of success of a single run. However, the use of best value, mean and variance present some difficulties. In fact, the distribution of the best values is not Gaussian. Therefore, the distance between the best and the mean values, or the value of the variance in general, does not give an exact indication of the repeatability of the result. Moreover, it changes during the process; therefore we cannot define a priori the required number of runs to produce

a correct estimation of mean and variance. In addition to that, the minimum number of samples that are required to have a sufficient statistics is not well defined for space problems. Note that the use of the best value could be misleading since, statistically, even a simple random sampling can converge to the global optimum. On the other hand, an algorithm converging, on average, to a good value with a small variance does not guarantee that it will be able to find the best possible solution. For example, given the integer numbers between 0 and 10, let us assume that we want to find the minimum one, i.e. 0, and that an algorithm returns 50% of the times the value 0 and 50% of the times the value 10. The mean would be 5 and the variance about 28, which would lead to the conclusion that the value 0 can be found with probability 0.014 under the assumption of Gaussian distribution. This conclusion is clearly wrong. Assume now that an algorithm returns solutions with mean value equal to 5 and variance equal to 30 and another algorithm returns solutions with mean equal to 3 and variance equal to 10. In this case, which of the two algorithms is better performing is not well defined because the algorithm with higher mean value has a higher variance and, thus, also a higher probability to generate solutions better than the average one. Then, if the distribution that describes the statistical phenomenon is not known, these two numbers are not sufficient to claim that one algorithm is better than the other. Note that, because of this evidence, statistical tests, like the *t-test*, that start from the assumption of a Gauss distribution, are not applicable or provide unreliable results.

An alternative index that can be used to assess the effectiveness of a stochastic algorithm is the success rate p_s, which is related to j_s in Algorithm D.2 by $p_s = j_s / n$. Considering the success as the referring index for a comparative assessment implies two main advantages. First, it gives an immediate and unique indication of the algorithm effectiveness, addressing all the issues highlighted above, and second, the success rate can be represented with a binomial probability density function (pdf), independent of the number of function evaluations, the problem and the type of optimisation algorithm. This latter characteristic implies that the test can be designed fixing a priori the number of runs *n*, on the basis of the error we can accept on the estimation of the success rate. We propose to use a statistical theory developed by Minisci et al. [123]. It is to assume that the sample proportion p_s of successes (the success rate for a given *n* in our case) can be approximated with a normal distribution, i.e. $p_s \sim N_p\left\{\theta_p, \theta_p(1-\theta_p)/n\right\}$, where θ_p is the unknown true proportion of successes, and that the probability of p_s to be at distance d_{err} from θ_p, $P_r\left[\left|p_s - \theta_p\right| \le d_{err} \mid \theta_p\right]$ is at least $1-\alpha_b$ (see Ref. [124]). This leads to the expression:

$$n \ge \theta_p(1-\theta_p)\chi^2_{(1),\alpha_b} \Big/ d^2_{err} \tag{D.3}$$

and to the conservative rule:

$$n \ge 0.25\chi^2_{(1),\alpha_b} \Big/ d^2_{err} \tag{D.4}$$

obtained if $\theta_p = 0.5$. For our tests we required an error $d_{err} \leq 0.06$ with a 92% confidence ($\alpha_b = 0.08$), which, according to Eq. (D.4), yields $n \geq 94$. This was extended to $n_{runs} = 100$ for all the tests in this chapter in order to have a higher confidence in the result. In order to have a feeling of the speed of convergence, stochastic based methods were applied to the solution of the whole problem for an increasing number of function evaluations.

For the incremental approach, we defined a number of performance indicators that aim at establishing if the reduction of the search space operated during the incremental search is reliable and efficient. It is here important to remind that the aim of the incremental approach is not to generate optimal solutions but to generate a set of sub-domains $\bar{D}_j \subseteq D$ bounding sets of locally optimal solutions. Therefore, the following indicators aim at measuring the ability of the incremental approach to repeatedly generate a tight enclosure of good solutions. Ideally, a good pruning would always yield few, small boxes enclosing the global optimum together with all the solutions satisfying $\delta_f < tol_f$.

Appendix E

TISSERAND PLANE

The Tisserand plane is another name for a plot having the orbital period P on the x axis and the radius of the pericentre r_π on the y axis. This plane turned out to be a very useful tool to design MGA trajectories in a certain planetary system [36].

In order to exploit the Tisserand plane for studying MGA solutions, first of all we need to approximate the full three-dimensional problem as a planar problem, thus neglecting the out-of plane components of all the orbits involved (bodies and spacecraft). We will also consider circular all the orbits of the planets in the planetary system (but not the one of the spacecraft). These approximations, nevertheless, are well met for the Solar System (with the exception of Pluto and Mercury) and for Ganymede and Callisto around Jupiter (the case under consideration).

We decide not to deal with the position of the orbit in space, which for a planar problem is represented only by the anomaly of the pericentre. The phasing is neglected, too, which means that we assume that bodies are in the correct position of their orbits whenever needed. This further assumption allows removing completely the time from the problem, and dealing only with geometrical orbital intersections. The shape of any interplanetary transfer orbit is fully determined, under these assumptions, by two parameters: the period and the radius of pericentre will be used.

By considering a plot with radius of pericentre and period, each transfer orbit is represented by one specific point in this plane (i.e., the so-called Tisserand plane). It can also be stated that performing a swing-by of a given planet changes the period of the orbit and its radius of the pericentre, leaving unchanged the relative velocity (v_∞) with respect to the same planet. The idea is then to consider the locus of orbits which intersect a planet with the same relative velocity, and draw the locus as an iso-v_∞ line in the Tisserand plane. Given a spacecraft orbit, in terms of P and r_π, and thus a point in the Tisserand plane, it is possible to compute the v_∞ with respect to a given planet. Then the correspondent iso-v_∞ curve can be drawn in the Tisserand plane: each swing-by with the planet will change the orbital period and

the radius of the pericentre such to move the point on the plane in either direction of the same iso-v_∞ curve. Several iso-v_∞ curves can be drawn, relative to a planet, for different values of v_∞. In addition, several planets can be considered: in this way, it is possible to visualise graphically how a swing-by of one planet can change the period, the radius of the pericentre, and the relative velocity with respect to *another* planet.

Fig. E.1 shows a Tisserand plane for Ganymede and Callisto around Jupiter. Iso-v_∞ lines for different values of v_∞ are shown in red. The blue lines represent the envelope of the iso-v_∞ lines of corresponding planets. On the left of this line, or on the top of it, there is no intersection between the spacecraft orbit and the planet orbit, and thus no swing-by is possible.

A MGA trajectory can be represented as a path in the Tisserand plane, moving from a starting point to a given target point or set of points. If the trajectory is purely ballistic, then the path will always move along one of the iso-v_∞ lines: in fact, only swing-bys are available to change P and r_π. When DSMs are considered, then the Δv given by the manoeuvre allows any displacement in the Tisserand plane, without any particular constraint. It is worth noticing, though, that a tangential thrust at pericentre does not modify the pericentre of the orbit itself, but only the period, thus moving along the y axis in the plane.

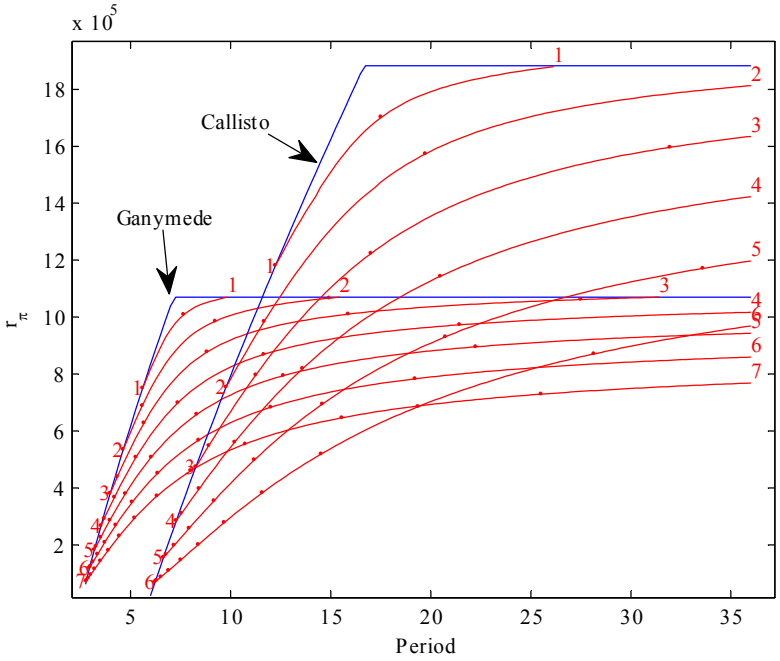


Fig. E.1. Tisserand plane for Ganymede and Callisto around Jupiter. Iso-v_∞ lines (red) for the two bodies are represented for different values of v_∞, specified in km/s. The dots along each single line are spaced by a swing-by in which the radius of pericentre is 1.1 radii of the planet: usually this is the lowest allowed value, so two consecutive dots represent the maximum displacement achievable through a swing-by.

REFERENCES

1. J. Gribbin, “Science: A history 1543 - 2001”, Penguin, 2002. ISBN: 978-0140297416
2. K. E. Tsiolkovsky, “The exploration of cosmic space by means of reaction revices (Russian: Исследование мировых пространств реактивными приборами)”, 1903.
3. G. P. Kennedy, “Germany's V-2 Rocket”, Schiffer Publishing Ltd, 2006. ISBN: 978-0764324529
4. C. Burgess, C. Dubbs, “Animals in space: From research rockets to the space shuttle”, *Space Exploration*, ed. J. Mason, Springer, 2007. ISBN: 978-0387360539
5. P. Bizony, J. Doran, “Starman: The truth behind the legend of Yuri Gagarin ”, Bloomsbury Publishing PLC, 1999. ISBN: 978-0747542674
6. A. Chaikin, “A man on the Moon: The voyages of the Apollo astronauts”, Penguin, 1998. ISBN: 978-0143112358
7. NASA/JPL, “Mariner-Venus 1962 final project report”, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19660005413_1966005413.pdf, 1965.
8. R. Frauenholz, B., D. L. Farless, “Pioneer 10/11 mission results”, in *Proceedings of AIAA Mechanics and Control of Flight Conference*, Anaheim, Calif., USA, 1974.
9. D. L. Shirley, “Mariner 10 mission analysis - Application of a black art”, in *Proceedings of American Institute of Aeronautics and Astronautics, Aerospace Sciences Meeting*, Pasadena, California, USA, 1975.
10. R. Shotwell, “Phoenix - The first Mars Scout mission”, in *Proceedings of 55th International Astronautical Congress (IAC 2004)*, Vancouver, Canada, 2004.
11. D. A. Spencer, J. L. Bell, G. Beutelschies, R. A. Mase, J. C. Smith, “2001 Mars Odyssey mission design”, in *Proceedings of AAS/AIAA Astodynamics Conference*, Quebec City, Que., Canada, 2002.
12. R. Schmidt, J. D. Credland, A. Chicarro, P. Moulinier, “ESA's Mars express mission - Europe on its way to Mars”, *European Space Agency Bulletin*, n. 98, p. 56-66, 1999.
13. R. A. Cook, “The Mars exploration rover project”, *Acta Astronautica*, vol. 57, n. 2-8, p. 116-120. DOI: 10.1016/j.actaastro.2005.03.008
14. J. Vago, B. Gardini, G. Kminek, P. Baglioni, G. Gianfiglio, et al., “ExoMars: Searching for life on the red planet”, *European Space Agency Bulletin*, n. 126, p. 16-23, 2006.

15. V. Badescu (ed.), "Mars: Prospective energy and material resources", Springer, 2010. ISBN: 978-3642036286
16. D. F. Landau, J. M. Longuski, "Comparative assessment of human-Mars-mission technologies and architectures", *Acta Astronautica*, vol. 65, n. 7-8, p. 893-911, 2009. DOI: 10.1016/j.actaastro.2009.02.008
17. M. A. Minovitch, "The invention that opened the solar system to exploration", *Planetary and Space Science*, vol. 58, n. 6, p. 885-892, 2010. DOI: 10.1016/j.pss.2010.01.008
18. C. R. McInnes, "Solar Sailing: Technology, Dynamics and Mission Applications", *Springer Praxis Books - Astronomy and Planetary Sciences* Springer, 1999. ISBN: 978-3-540-21062-7
19. E. C. Kohlhase, P. A. Penzo, "Voyager mission description", *Space science reviews*, vol. 21, n. 2, p. 77-101, 1977.
20. J. M. Longuski, S. N. Williams, "Automated design of gravity-assist trajectories to Mars and outer planets", *Celestial Mechanics and Dynamical Astronomy*, vol. 52, n. 3, p. 207-220, 1991.
21. R. E. Diehl, D. I. Kaplan, P. A. Penzo, "Satellite tour design for the Galileo mission", in *Proceedings of American Institute of Aeronautics and Astronautics, Aerospace Sciences Meeting*, Reno, NV, USA, 1983.
22. F. Peralta, S. Flanagan, "Cassini interplanetary trajectory design", *Control Engineering Practice*, vol. 3, n. 11, p. 1603-10, 1995. DOI: 10.1016/0967-0661(95)00171-P
23. D. L. Matson, J.-P. Lebreton, L. J. Spilker, "The Cassini-Huygens mission to the Saturnian system", in *Proceedings of International Conference TITAN - From Discovery to Encounter, April 13, 2004 - April 17, 2004*, Noordwijk, Netherlands, 2004.
24. A. A. Wolf, J. C. Smith, "Design of the Cassini tour trajectory in the Saturnian system", *Control Engineering Practice*, vol. 3, n. 11, p. 1611-19, 1995. DOI: 10.1016/0967-0661(95)00172-Q
25. J. V. McAdams, D. W. Dunham, R. W. Farquhar, A. H. Taylor, B. G. Williams, "Trajectory design and maneuver strategy for the MESSENGER mission to Mercury", *Journal of Spacecraft and Rockets*, vol. 43, n. 5, p. 1054-1064, 2006.
26. J. V. McAdams, "MESSENGER mission overview and trajectory design", *Advances in the Astronautical Sciences*, vol. 116, n. 1, p. 643-662, 2003.
27. J. Schoenmaekers, R. Bauske, "Re-design of the rosetta mission for launch in 2004", in *Proceedings of 18th International Symposium on Space Flight Dynamics, October 11, 2004 - October 15, 2004*, Munich, Germany, 2004.
28. Y. Guo, R. W. Farquhar, "New Horizons mission design for the Pluto-Kuiper Belt mission", in *Proceedings of AIAA/AAS Astrodynamics Specialist Conference*, Monterey, CA, USA, 2002.
29. R. Marsden, N. Angold, "The epic voyage of ulysses", *European Space Agency Bulletin*, vol. 2008, n. 136, p. 3-7, 2008.
30. S. Campagnola, C. Corral, R. Jehn, A. Yàñez, "BepiColombo Mercury cornerstone mission analysis: inputs for the reassessment phase of the

definition study", ESA-ESOC Mission Analysis Office, MAO Working Paper No. 452, Darmstadt, 2003.

31. D. Garcia Yárnoz, P. De Pascale, R. Jehn, S. Campagnola, C. Corral, et al., "BepiColombo Mercury cornerstone consolidated report on mission analysis", ESA-ESOC Mission Analysis Office, MAO Working Paper No. 466, Darmstadt, 2006.
32. D. Garcia Yárnoz, R. Jehn, M. Croon, "Interplanetary navigation along the low-thrust trajectory of BepiColombo", *Acta Astronautica*, vol. 59, n. 1-5, p. 284-293, 2006. DOI: 10.1016/j.actaastro.2006.02.028
33. D. Garcia Yárnoz, J. Schoenmaekers, R. Jehn, "BepiColombo Mercury cornerstone - mission analysis: chemical options in 2014 and 2015", ESA-ESOC Mission Analysis Section, MAS Working Paper No. 526, Darmstadt, 2008.
34. R. Jehn, S. Campagnola, D. Garcia Yárnoz, A. Yanez, "BepiColombo Mercury cornerstone - Mission Analysis: chemical options", ESA-ESOC Mission Analysis Office, MAO Working Paper No. 486, Darmstadt, 2005.
35. Joint Jupiter Science Definition Team, NASA/ESA Study Team, "EJSM NASA/ESA joint summary report", NASA/ESA, http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=44035#, 2009.
36. A. V. Labunsky, O. V. Papkov, K. G. Sukhanov, "Multiple gravity assist interplanetary trajectories", *Earth Space Institute Book Series*, Gordon and Breach Science Publishers, 1998. ISBN: 90-5699-090-X
37. N. J. Strange, J. M. Longuski, "Graphical method for gravity-assist trajectory design", *Journal of Spacecraft and Rockets*, vol. 39, n. 1, p. 9-16, 2002.
38. A. E. Bryson, Y.-C. Ho, "Applied optimal control", Taylor & Francis, New York, 1975. ISBN: 978-0891162285
39. P. Di Lizia, G. Radice, "Advanced global optimisation tools for mission analysis and design", European Space Agency, Advanced Concepts Team, http://www.esa.int/gsp/ACT/ariadna/completed.htm#MA, 2004.
40. P. Di Lizia, G. Radice, D. Izzo, M. Vasile, "On the solution of interplanetary trajectory design problems by global optimisation methods", in *Proceedings of Global Optimisation 2005*, Almeria, Spain, 2005.
41. M. Vasile, P. De Pascale, "Preliminary design of multiple gravity-assist trajectories", *Journal of Spacecraft and Rockets*, vol. 43, n. 4, p. 794-805, 2006.
42. B. Dachwald, "Optimization of Interplanetary Solar Sailcraft Trajectories Using Evolutionary Neurocontrol", *Journal of Guidance, Control, and Dynamics*, vol. 27, n. 1, p. 66-72, 2004.
43. J. W. Hartmann, V. L. Coverstone-Carroll, S. N. Williams, "Optimal interplanetary spacecraft trajectories via a Pareto genetic algorithm", *Journal of the Astronautical Sciences*, vol. 46, n. 3, p. 267-282, 1998.
44. P. Gurfil, N. J. Kasdin, "Niching genetic algorithms-based characterization of geocentric orbits in the 3D elliptic restricted three-body problem", *Computer Methods in Applied Mechanics and Engineering*, vol. 191, n. 49-50, p. 5683-5706, 2002. DOI: 10.1016/S0045-7825(02)00481-4

45. S. M. Pessina, S. Campagnola, M. Vasile, "Preliminary analysis of interplanetary trajectories with aerogravity and gravity assist manoeuvres", in *Proceedings of 54th International Astronautical Congress*, Bremen, Germany, 2003.
46. P. Rogata, E. Di Sotto, M. Graziano, F. Graziani, "Guess value for interplanetary transfer design through genetic algorithms", in *Proceedings of 13th AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico, 2003.
47. M. Vasile, R. Biesbroek, L. Summerer, A. Galvez, G. Kminek, "Options for a mission to Pluto and beyond", in *Proceedings of 13th AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico, 2003.
48. G. Hughes, C. R. McInnes, "Solar Sail Hybrid Trajectory Optimisation", *Advances in the Astronautical Sciences*, vol. 109p. 2369-2380, 2001.
49. B. Dachwald, "Optimisation of solar sail interplanetary trajectories using evolutionary neurocontrol", *Journal of Guidance, Control, and Dynamics*, vol. 27, n. 1, p. 66-72, 2004.
50. D. J. Wirthman, S. Y. Park, S. R. Vadali, "Trajectory optimisation using parallel shooting method on parallel computer", *Journal of Guidance, Control, and Dynamics*, vol. 18, n. 2, p. 377-379, 1995.
51. J. T. Betts, S. O. Erb, "Optimal low thrust trajectories to the Moon", *SIAM Journal of Applied Dynamical Systems*, vol. 2, n. 2, p. 144-170, 2003.
52. M. Vasile, "A global approach to optimal space trajectory design", *Advances in the Astronautical Sciences*, vol. 114, n. SUPPL, p. 621-640, 2003.
53. M. Rosa Sentinella, "Comparison and integrated use of differential evolution and genetic algorithms for space trajectory optimisation", in *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, 2007. DOI: 10.1109/CEC.2007.4424575
54. K. V. Price, R. M. Storn, J. A. Lampinen, "Differential evolution: a practical approach to global optimization", *Natural computing series*, ed. G. Rozenberg, Springer, Berlin, 2005. ISBN: 978-3540209508
55. R. Storn, K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, vol. 11p. 341-359, 1997.
56. M. Vasile, E. Minisci, M. Locatelli, "A dynamical system perspective on evolutionary heuristics applied to space trajectory optimization problems", in *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009.
57. J. Kennedy, R. Eberhart, "Particle swarm optimization", in *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, 1995. DOI: 10.1109/ICNN.1995.488968
58. R. H. Leary, "Global Optimization on Funneling Landscapes", *Journal of Global Optimization*, vol. 18, n. 4, p. 367-383, 2000. DOI: 10.1023/A:1026500301312
59. D. R. Myatt, V. M. Becerra, S. J. Nasuto, J. M. Bishop, "Advanced global optimisation for mission analysis and design", European Space Agency,

Advanced Concepts Team, Ariadna Final Report 03-4101a, http://www.esa.int/gsp/ACT/ariadna/completed.htm#MA, 2004.

60. D. Izzo, "1st ACT global trajectory optimisation competition: Problem description and summary of the results", *Acta Astronautica*, vol. 61, n. 9, p. 731-734, 2007.
61. V. M. Becerra, D. R. Myatt, S. J. Nasuto, J. M. Bishop, D. Izzo, "An efficient pruning technique for the global optimisation of multiple gravity assist trajectories", in *Proceedings of International Workshop on Global Optimization, GO 2005*, Almeria, Spain, 2005.
62. A. D. Olds, C. A. Kluever, M. L. Cupples, "Interplanetary mission design using differential evolution", *Journal of Spacecraft and Rockets*, vol. 44, n. 5, p. 1060-1070, 2007. DOI: 10.2514/1.27242
63. D. M. Pisarevsky, P. Gurfil, "A Memetic Algorithm for Optimizing High-Inclination Multiple Gravity-Assist Orbits", in *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009.
64. M. Rosa Sentinella, L. Casalino, "Genetic algorithm and indirect method coupling for low-thrust trajectory optimization", in *Proceedings of 42th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Sacramento, California, 2006.
65. M. Vasile, E. Minisci, M. Locatelli, "On testing global optimization algorithms for space trajectory design", in *Proceedings of AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Honolulu, Hawaii, 2008.
66. M. Vasile, M. Locatelli, "A hybrid multiagent approach for global trajectory optimization", *Journal of Global Optimization*, 2008. DOI: 10.1007/s10898-008-9329-3
67. A. E. Petropoulos, T. D. Kowalkowski, M. A. Vavrina, D. W. Parcher, P. A. Finlayson, et al., "1st ACT global trajectory optimisation competition: results found at the Jet Propulsion Laboratory", *Acta Astronautica*, vol. 61, n. 9, p. 806-815, 2007. DOI: 10.1016/j.actaastro.2007.03.013
68. M. Belló Mora, J. L. Cano, "1st ACT Global Trajectory Optimisation Competition: results found at DEIMOS Space", *Acta Astronautica*, vol. 61, n. 9, p. 794-805, 2007. DOI: 10.1016/j.actaastro.2007.03.007
69. M. Vasile, S. Campagnola, "Design of low-thrust multi-gravity assist trajectories to Europa", *Journal of the British Interplanetary Society*, vol. 62, n. 1, p. 15-31, 2009.
70. J. A. Sims, A. J. Staugler, J. M. Longuski, "Trajectory options to Pluto via gravity assists from Venus, Mars, and Jupiter", *Journal of Spacecraft and Rockets*, vol. 34, n. 3, p. 347-353, 1997.
71. M. R. Patel, J. M. Longuski, "Automated design of Delta-V gravity-assist trajectories for solar system exploration", in *Proceedings of AAS/AIAA Astrodynamics Conference*, Victoria, BC, Can, 1993.
72. J. A. Sims, J. M. Longuski, A. J. Staugler, "V∞ leveraging for interplanetary missions: multiple-revolution orbit techniques", *Journal of Guidance, Control, and Dynamics*, vol. 20, n. 3, p. 409-415, 1997.

73. A. F. Heaton, N. J. Strange, J. M. Longuski, E. P. Bonfiglio, "Automated design of the Europa Orbiter tour", *Journal of Spacecraft and Rockets*, vol. 39, n. 1, p. 17-22, 2002.
74. A. E. Petropoulos, J. M. Longuski, E. P. Bonfiglio, "Trajectories to Jupiter via gravity assists from Venus, Earth, and Mars", *Journal of Spacecraft and Rockets*, vol. 37, n. 6, p. 776-783, 2000.
75. A. E. Petropoulos, J. M. Longuski, "Shape-based algorithm for automated design of low-thrust, gravity-assist trajectories", *Journal of Spacecraft and Rockets*, vol. 41, n. 5, p. 787-796, 2004.
76. T. T. McConaghy, T. J. Debban, A. E. Petropoulos, J. M. Longuski, "Design and optimization of low-thrust trajectories with gravity assists", *Journal of Spacecraft and Rockets*, vol. 40, n. 3, p. 380-387, 2003.
77. M. Vasile, F. Bernelli Zazzera, "Optimizing low-thrust and gravity assist maneuvers to design interplanetary trajectories", *The Journal of the Astronautical Sciences*, vol. 51, n. 1, 2003.
78. C. Blum, M. J. Blesa Aguilera, A. Roli, M. Sampels (eds.), "Hybrid metaheuristics, an emerging approach to optimization", *Studies in Computational Intelligence*, J. Kacprzyk ed., Springer, 2008. ISBN: 978-3-540-78294-0
79. D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, J. M. Bishop, "Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories", *Journal of Global Optimization*, 2006.
80. P. De Pascale, M. Vasile, S. Casotto, "Preliminary analysis of low-thrust gravity assist trajectories by an inverse method and a global optimization technique", in *Proceedings of 18th International Symposium on Space Flight Dynamics*, Munich, Germany, 2004.
81. I. M. Ross, C. N. D'Souza, "Hybrid optimal control framework for mission planning", *Journal of Guidance, Control, and Dynamics*, vol. 28, n. 4, p. 686-697, 2005.
82. O. Von Stryk, M. Glocker, "Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation", in *Proceedings of ADPM 2000 – The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, Dortmund, Germany, 2000.
83. B. J. Wall, B. A. Conway, "Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics", *Journal of Global Optimization*, vol. 44, n. 4, p. 493-508, 2009. DOI: 10.1007/s10898-008-9352-4
84. C. M. Chilana, B. A. Conway, "Using Genetic Algorithms for the Construction of a Space Mission Automaton", in *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009.
85. A. E. Rizzoli, R. Montemanni, E. Lucibello, L. M. Gambardella, "Ant colony optimization for real-world vehicle routing problems - From theory to applications", *Swarm Intelligence*, vol. 1, n. 2, p. 135-151, 2007. DOI: 10.1007/s11721-007-0005-x

86. M. Dorigo, L. M. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem”, *IEEE Transactions on Evolutionary Computation*, vol. 1, n. 1, p. 53-66, 1997. DOI: 10.1109/4235.585892
87. M. Dorigo, T. Stützle, “Ant colony optimization”, The MIT Press, Cambridge, Massachusetts, 2004. ISBN: 0-262-04219-3
88. D. Merkle, M. Middendorf, H. Schmeck, “Ant colony optimization for resource-constrained project scheduling”, *IEEE Transactions on Evolutionary Computation*, vol. 6, n. 4, p. 333-346, 2002.
89. C. Blum, “Beam-ACO - Hybridizing ant colony optimization with beam search: An application to open shop scheduling”, *Computers and Operations Research*, vol. 32, n. 6, p. 1565-1591, 2005.
90. V. P. Eswaramurthy, A. Tamilarasi, “Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems”, *International Journal of Advanced Manufacturing Technology*, vol. 40, n. 9-10, p. 1004-1015, 2009. DOI: 10.1007/s00170-008-1404-x
91. K.-L. Huang, C.-J. Liao, “Ant colony optimization combined with taboo search for the job shop scheduling problem”, *Computers and Operations Research*, vol. 35, n. 4, p. 1030-1046, 2008. DOI: 10.1016/j.cor.2006.07.003
92. A. N. Sinha, N. Das, G. Sahoo, “Ant colony based hybrid optimization for data clustering”, *Kybernetes*, vol. 36, n. 2, p. 175-191, 2007. DOI: 10.1108/03684920710741215
93. M. H. Kaplan, “Modern spacecraft dynamics & control”, John Wiley & Sons, New York, 1976. ISBN: 9780471457039
94. D. A. Vallado, “Fundamentals of astrodynamics and applications (Second edition)”, *Space Technology Library*, Microcosm Press/Kluwer Academic Publishers, El Segundo, CA, USA/Dordrecht, The Netherlands, 2001. ISBN: 978-11881883128
95. R. H. Battin, “An introduction to the mathematics and methods of astrodynamics”, Revised edition, *AIAA Education Series*, AIAA, New York, 1999. ISBN: 1-56347-342-9
96. H. Shen, P. Tsiotras, “Using Battin's method to obtain multiple-revolution Lambert's solutions”, in *Proceedings of AAS/AIAA Astrodynamics Specialist Conference*, Big Sky, Montana, USA, 2003.
97. K. J. Ball, G. F. Osborne, “Space vehicle dynamics”, Oxford University (Clarendon) Press, 1967.
98. C. H. Acton, “Ancillary Data Services of NASA's Navigation and Ancillary Information Facility”, *Planetary and Space Science*, vol. 44, n. 1, p. 65-70, 1996.
99. D. Izzo, “Advances in global optimisation for space trajectory design”, in *Proceedings of 25th International Symposium on Space Technology and Science*, 2006.
100. L. A. D'Amario, “Galileo trajectory design”, *Space science reviews*, vol. 60p. 23-78, 1992.

101. E. W. Weisstein, “Sphere point picking”, available from: http://mathworld.wolfram.com/SpherePointPicking.html, cited 30 August 2007.
102. M. Ceriotti, M. Vasile, C. Bombardelli, “An incremental algorithm for fast optimisation of multiple gravity assist trajectories”, in *Proceedings of 58th International Astronautical Congress*, Hyderabad, India, 2007.
103. H. Oberth, “Ways to spaceflight (translation of "Wege zur Raumschiffahrt")”, R. Oldenbourg Verlag, Munich-Berlin, 1929.
104. M. Vasile, M. Ceriotti, G. Radice, V. M. Becerra, S. J. Nasuto, et al., “Global trajectory optimisation: can we prune the solution space when considering deep space manoeuvres?”, European Space Agency, Advanced Concepts Team, http://www.esa.int/gsp/ACT/ariadna/completed.htm#MA, 2008.
105. V. M. Becerra, S. J. Nasuto, J. D. Anderson, M. Ceriotti, C. Bombardelli, “Search space pruning and global optimization of multiple gravity assist trajectories with deep space manoeuvres”, in *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, 2007.
106. J. T. Betts, “Practical methods for optimal control using nonlinear programming”, *Advances in Design and Control*, Society for Industrial & Applied Mathematics, 2001. ISBN: 978-0898714883
107. D. H. Wolpert, W. G. Macready, “No free lunch theorems for optimization”, *IEEE Transactions on Evolutionary Computation*, vol. 1, n. 1, p. 67-82, 1997.
108. R. Bellman, “Dynamic Programming”, Princeton University Press, Princeton, NJ, USA, 1957.
109. D. R. Jones, “A taxonomy of global optimization methods based on response surfaces”, *Journal of Global Optimization*, vol. 21p. 345-383, 2001.
110. W. Tu, R. W. Mayne, “Studies of multi-start clustering for global optimization”, *International Journal for Numerical Methods in Engineering*, vol. 53, n. 9, p. 2239-2252, 2002.
111. D. J. Wales, J. P. K. Doye, “Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms”, *The Journal of Physical Chemistry A*, vol. 101, n. 28, p. 5111-5116, 1997. DOI: 10.1021/jp970984n
112. B. Addis, M. Locatelli, F. Schoen, “Local optima smoothing for global optimization”, *Optimization Methods and Software*, vol. 20, n. 4-5, p. 417-437, 2005.
113. D. Comaniciu, P. Meer, “Mean shift: a robust approach toward feature space analysis”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, n. 5, p. 603-619, 2002. DOI: 10.1109/34.1000236
114. D. R. Jones, C. D. Perttunen, B. E. Stuckman, “Lipschitzian optimization without the Lipschitz constant”, *Journal of Optimization Theory and Applications*, vol. 79, n. 1, p. 157-181, 1993.

115. W. Huyer, A. Neumaier, "Global optimization by multilevel coordinate search", *Journal of Global Optimization*, vol. 14, n. 4, p. 331-355, 1999. DOI: 10.1023/A:1008382309369
116. ESA, "Laplace mission summary", available from: http://sci.esa.int/science-e/www/area/index.cfm?fareaid=107, cited 9 September 2008.
117. Laplace CDF Team, "Laplace mission analysis, kick off", PowerPoint presentation, Darmstadt, Germany, 21 May 2008.
118. E. Canalias, A. Boutonnet, "Interplanetary trajectory preliminary design tool for the Jovian Minisat Explorer", in *Proceedings of Nonlinear Science and Complexity (NSC 2008)*, Porto, Portugal, 2008.
119. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical recipes in C: the art of scientific computing", Second edition, Cambridge University Press, 1992. ISBN: 0521437148
120. D. E. Goldberg, "Genetic algorithms in search, optimization & machine learning", Addison Wesley, Boston, MA, USA, 1989. ISBN: 978-0201157673
121. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II", in *Proceedings of 6th International Conference on Parallel Problem Solving from Nature, PPSN VI*, Paris, France, 2000.
122. B. Birge, "PSOt - a particle swarm optimization toolbox for use with Matlab", in *Proceedings of Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, 2003. DOI: 10.1109/SIS.2003.1202265
123. E. A. Minisci, G. Avanzini, "Orbit Transfer Manoeuvres as a Test Benchmark for Comparison Metrics of Evolutionary Algorithms", in *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009.
124. C. J. Adcock, "Sample size determination: a review", *The Statistician*, vol. 46, n. 2, p. 261-283, 1997.